NO. 35 $2.00 APRIL 1981

MICRO™

THE 6502 JOURNAL

Atari Error Messages

MEMSEARCH for AIM 65

BASIC Program Converter Between SYM and KIM

Print Using for the PET

Apple Memory Maps

Joysticks for the OSI C4

The Perfect Fit

The Micromodem II data communications system and the Apple II* computer. What better combination to maximize the capabilities of your personal computer!

This popular direct connect modem can transmit data between an Apple II and another Apple II, a terminal, another microcomputer, minicomputer or even a large time-sharing computer anywhere in North America. The Micromodem II has unique automatic dialing and answer capabilities which further increases the communications possibilities between the Apple II and another computer or terminal.

You can send and/or receive messages or data when you are out of your office, home or out of town. Your branch business locations can communicate with each other regarding inventory and other matters over the phone. Or you can communicate with friends across the country. And you can access information utilities like the SOURCE for various business and personal applications.

The Micromodem II consists of two parts. One part includes the printed circuit board which holds the Micromodem II, ROM firmware and the serial interface. The board plugs directly into the Apple II providing all the functions of a serial interface card plus programmable auto dialing and auto answer capabilities. The on-board ROM firmware enables the Micromodem II to operate in any of three modes to perform different tasks-terminal mode, remote console and program control mode.

The other part of the Micromodem II datacomm system is a Microcoupler which connects the Micromodem board and Apple II to a telephone line. The Microcoupler gets a dial tone, dials numbers, answers the phone and hangs up when a transmission is over. There are none of the losses or distortions associated with acoustic couplers. The Microcoupler is compatible with any North American standard telephone lines and is FCC-approved for direct connection in the U.S. It works with standard dial phone service or Touch-tone service.

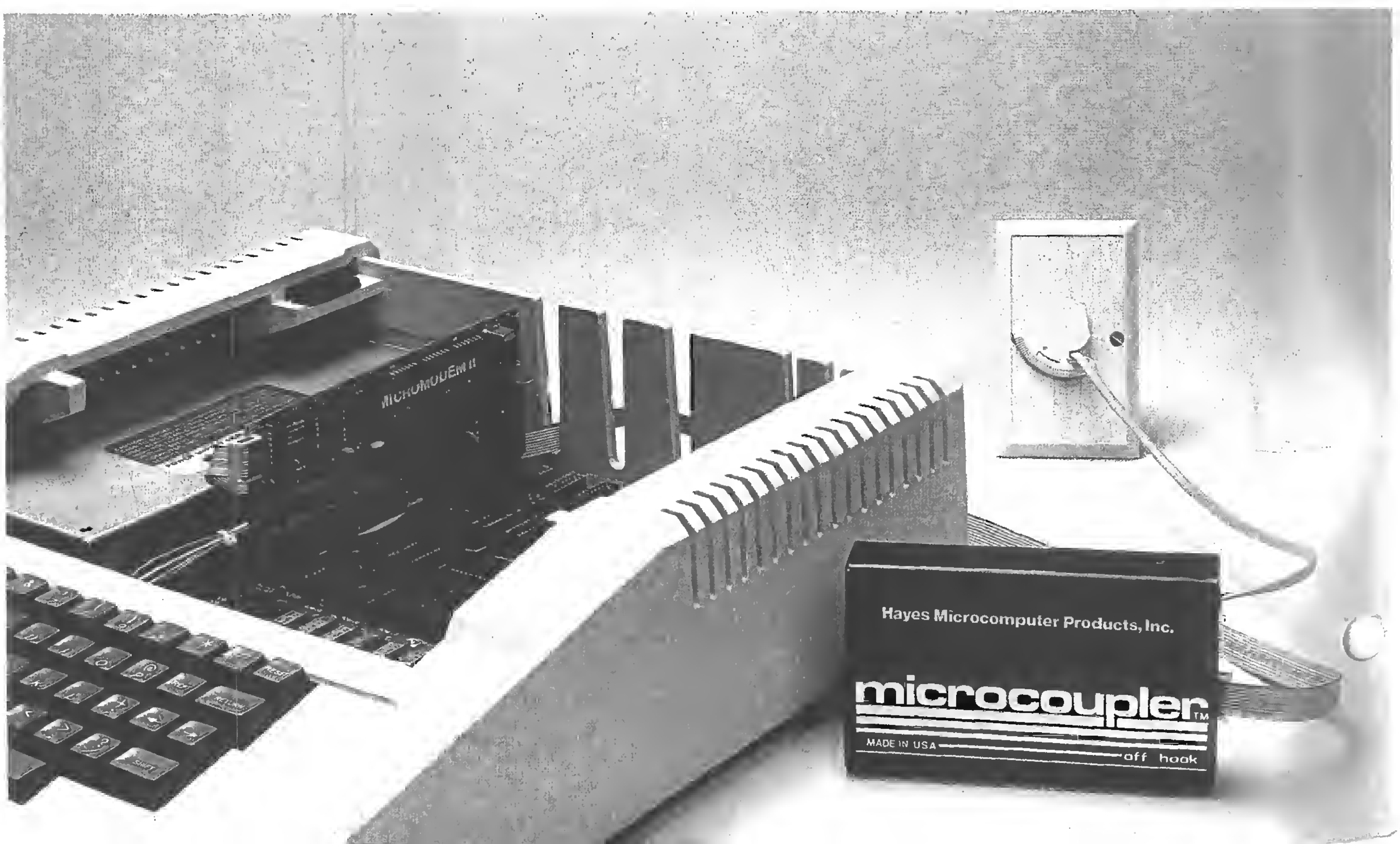
The Micromodem II is completely compatible with Bell 103-type modems. Full and half-duplex operating modes are available as well as speed selectable transmission rates of 110 and 300 bps.

Why not increase your Apple II's capabilities by outfitting it with the sophisticated Micromodem II data communications system? The Micromodem II is available at retail computer stores nationwide. For the store nearest you, call or write:

Hayes™

Hayes Microcomputer Products Inc.
5835 Peachtree Corners East, Norcross, Georgia 30092 (404) 449-8791

™ Micromodem II is a trademark of Hayes Microcomputer Products, Inc.
*Apple II is a registered trademark of Apple Computer Inc.
The Micromodem II can also be used with the Bell & Howell computer.



Our new Prodisk card
gives the
small
PROGRAMMA
HARDWARE
Up to 5 million bytes of on-line storage with 8" floppies. And, without stealing power from your Apple II*!
Your business is more complicated than anyone realizes. Or, you're growing faster than expected. Does that mean you have to trade in your reliable Apple II for a bigger, more costly system? Or, is there a simple, reasonable solution?
You bet there is, now that Programma International has introduced the Teeter Electronics PRODISK controller card.
PRODISK is like having the storage capability of ten Apples! Because with just four 8" floppy disk drives, its on-line storage capacity goes to a business-size 5 million bytes. Plus it delivers high-speed transfer of a half million bits per second.
With storage and speed like that you can really get a handle on your entire business. And, it won't interfere with your Apple's operation. You see, the new PRODISK card is powered from the drives, not the Apple. Since there's no significant power drain, other cards can be used with no problem.
Technically Inclined?
The card operates under Apple DOS 3.1 or 3.2, with 48K Apple II. It has full compatability with mini-disks. Handles from one to four 8" floppy drives. Single or double density disks use DMA transfer techniques . . . with high speed transfer of half million bits per second.
Priced Right
The amazing PRODISK controller card is priced at $645.00. And it's a tax deductible business expense. Its low price works out to be even lower! Same for the special Programma 8" floppy disk drives (800 or 850 Shugart equivalent). Example: two single sided drives, priced at $1549.00.
Get big business storage capacity for your small business right now . . . with PRODISK, available at your better computer stores, or direct from Programma International.
PRODISK
Big power for small business
PROGRAMMA
Programma International Inc.
2908 No. Naomi Street
Burbank, CA 91504
(213) 954-0240
e II and Apple DOS are trademarks of Apple Computer, Inc.

A BRILLIANT FUTURE FOR YOUR AIM-65 WITH THE BANKER MEMORY ™

Your 36K of free address space is the AIM's most valuable and limited resource. With today's large capacity RAM boards, ROM boards, disk systems, video boards, and other expansion accessories it is easy to deplete this resource before the application requirement is satisfied. MTU has solved this problem.

THE BANKER MEMORY contains 32K of RAM, 4 PROM sockets for 2716/2732/2332, a PROM programmer, 40 bits of parallel I/O, and 4 timers from two 6522 I/O chips. Addressing is extremely flexible with the RAM independently addressable in 4K blocks, PROM's independently addressable, and I/O addressable anywhere on a 64 byte boundary (even in AIM's I/O area at AXXX by adding a single jumper to the AIM).

This may sound familiar, but read on! Unlike other AIM compatible memory boards, THE BANKER MEMORY has on-board bank-switching logic! The four 8K blocks of RAM plus the 4 PROM sockets make up 8 **resources**, each associated with a bit in an Enable Register. Through this Enable Register resources may be turned on and off under software control. When a resource is off, its address space is freed for other uses. You can even put BANKER resources at the same address and switch among them for virtually unlimited RAM and PROM expansion! You can even have multiple page zero's and stacks! Do you need 160K byte of memory? It only takes 5 of THE BANKER MEMORY boards and you end up with 5 page zeros and stacks to boot!

There's more! The BANKER MEMORY also incorporates 18 bit addressing which allows for the 256K address spaces of the future. RAM, PROM, and I/O each has its own full 18 bit address decoder which allows these resources to be in different 64K banks. This board and other MTU products, such as our 320 by 200 dot VISIBLE MEMORY and Floppy Disk Controller with 16K DMA RAM, can turn your AIM into a truly powerful 6502 computer that far surpasses the packaged systems in functional performance.

INTRODUCTORY SPECIAL K-1032-1 32K BANKER MEMORY FULLY ASSEMBLED AND TESTED $395.00 ($450.00 as of March 1, 1980) or the K-1032-2 16K RAM only with bank switching and 18 bit address bus only $295.00

Isn't it time you took a closer look at MTU — we offer you power now with an eye to the future.

WRITE OR CALL TODAY FOR OUR 48 PAGE FALL 1980 6502 CATALOG
International requests include $1.00

VISA and MASTERCARD accepted

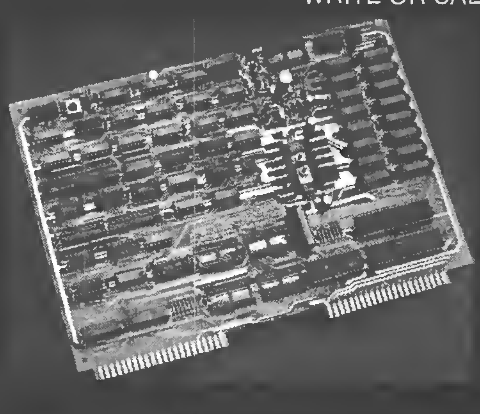


Photo credit:
SUPERNOVA CRAB NEBULA:
Palomar Observatory,
California Institute of Technology

MTU
Micro Technology Unlimited
2806 Hillsborough Street
P.O. Box 12106
Raleigh, NC 27605, U.S.A.
(919) 833-1458

NO. 35 APRIL 1981

MICRO™

THE 6502 JOURNAL

STAFF

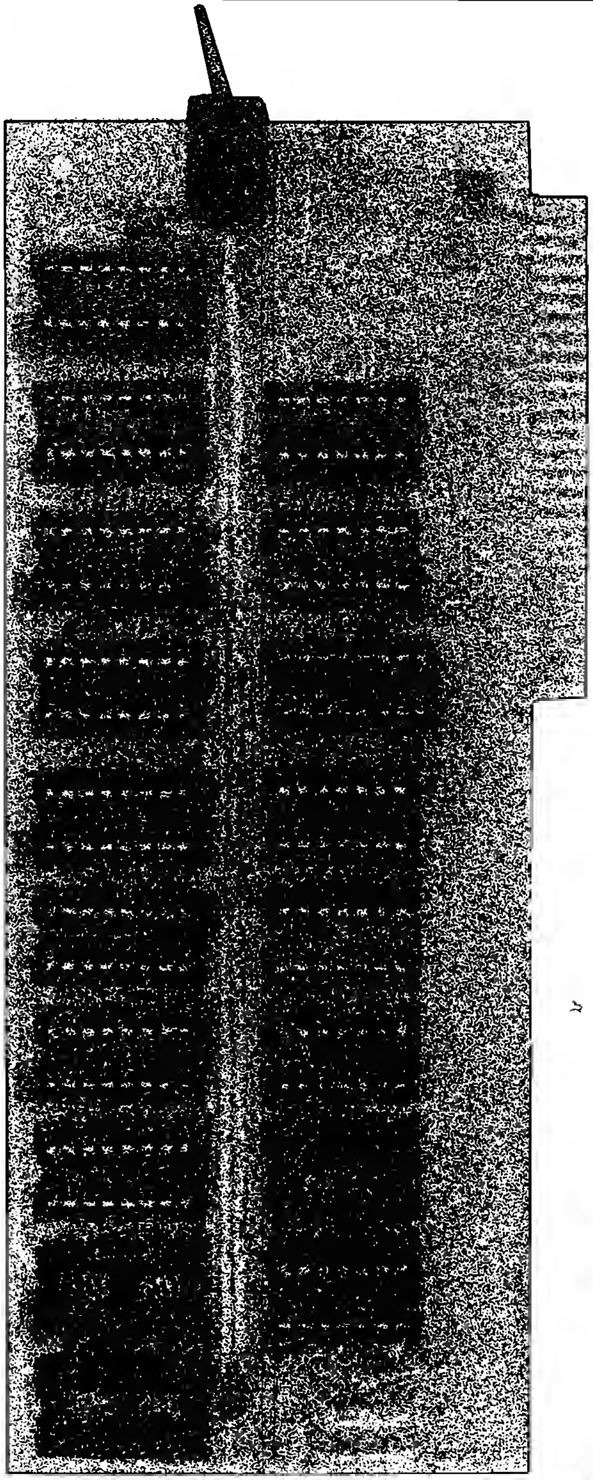
MICRO™ is published monthly by:
MICRO INK, Inc., Chelmsford, MA 01824
Second Class postage paid at:
Chelmsford, MA 01824 and additional offices
Publication Number: COTR 395770
ISSN: 0271-9002

Subscription Rates:	*Per Year*
U.S.	
through March 31	$15.00
thereafter	$18.00
Foreign surface mail	$21.00
Air mail:	
Europe	$36.00
Mexico, Central America	$39.00
Middle East, North Africa	$42.00
South America, Central Africa	$51.00
South Africa, Far East, Australasia	$60.00

For back issues, subscriptions, change of address or other information, write to:
MICRO
P.O. Box 6502
Chelmsford, MA 01824
or call
617/256-5515

CONTENTS

DEPARTMENTS



16K RAM Expansion Board for the Apple II*

$195.00

- expands your 48K Apple to 64K of programmable memory
- works with Microsoft Z-80 card, Visicalc, LISA ver 2.0 and other software
- eliminates the need for an Applesoft* or Integer Basic ROM Card
- switch selection of RAM or mother board ROM language
- includes installation and use manual
- fully assembled and tested

MasterCard

VISA

Visa and MasterCard accepted
Shipping and handling will be added unless the order is accompanied by a check or money order
N.C. residents add 4% sales tax

*Apple II and Applesoft are trademarks of Apple Computer, Inc.

ANDROMEDA

INCORPORATED**
P.O. Box 19144
Greensboro, NC 27410
(919) 852-1482

**Formerly Andromeda Computer Systems

MICRO Editorial

Staff and Stuff

Most readers probably never read the staff listing on the Table of Contents page, but those who do will note that since the beginning of this volume, (June 1980) we have added an associate publisher, special projects editor, art director, advertising manager, three micro specialists and a typesetter. This improved staffing permits MICRO to deliver a better product each month and to undertake a number of other projects as well.

One major undertaking is book publishing. A series of Apple books is under way, with the first, *MICRO/Apple, Volume 1*, to be released April 1. This collection of Apple articles from past issues of MICRO is intended for the beginner-to-intermediate Apple user. All of the material has been re-edited, re-typeset and many articles have been updated by the original authors and/or the MICRO staff. All programs have been re-entered, listed and tested. They are provided on a diskette which is an integral part of the book. The 224-page book is wire-bound and lies flat when open to make it easy to use.

Other books in the *MICRO/Apple* series will include reprints, original articles, new reference works, and more. This will permit us to present various types of material which do not work well in a magazine format: long articles or listings, good articles of limited scope, and so forth.

We are looking for additional material for other major microcomputers to support similar books for the PET, OSI, AIM, SYM, KIM and Atari. If you have material which you may not have submitted because you felt that it was not suited to a magazine presentation, please consider it for one of the books. If you have a complete manuscript for a 6502-based book, or even just the idea for one, please contact us. We may be interested in publishing it and distributing it to the 6502 world through our dealer network.

An Apple Solution

The February editorial addressed the problem of "Too Many Apples" —more Apple articles on a regular basis than we can incorporate in MICRO without overwhelming the other 6502-based microcomputers. The reader response may be summarized as:

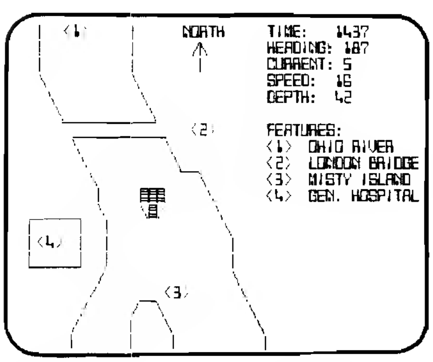
> no one favored "no change" or "print the extra Apple material in book form";
>
> only a few wanted to "print the best material without regard to microcomputer";
>
> more wanted to "publish an Apple supplement or quarterly" or "publish a monthly Apple magazine";
>
> most chose to "increase the size of MICRO to accommodate additional Apple material without reducing the coverage of the other micros."

This reader feedback and our in-house staff discussions agree, and the decision has been made to expand MICRO. Starting with the June 1981 issue, there will be "extra" pages devoted to Apple articles and advertisements. The number of pages will be determined partially by the extra advertising required to cover the additional production, printing and postage costs—without requiring an increase in either the single copy or subscription price. There will be at least 16 extra pages, and possibly 32 pages. This expansion will permit us to provide timely Apple coverage while maintaining our policy of serving the entire 6502 community.

Robert M. Tripp

Robert M. Tripp
Editor/Publisher

About the Cover—



Screen display on this month's cover — from the human point of view.

(Cover photo by Michael Rakip)

Cruising Down the River...

Imagine yourself cruising down the river on the Delta Queen. To navigate rivers you need maps and charts. Currently these are available as printed material, very detailed and very accurate. Accurate? Well, the chart was accurate when it was made, but how long ago was that? And what changes have occurred since then?

How about a computer displayable map —one that could be updated continuously by whatever authority has the charting responsibility, the U.S. Coast Guard or the U.S. Geodetic Survey? A diskette could be generated which contains the latest information along a particular route. It could provide whatever level of detail is required; from an overview as pictured on the cover, to the detail normally provided in navigational charts. As the position of the vessel is entered into the computer, manually in a simple system or automatically in a more advanced one, the display could change to provide the current map information.

In addition to the pure mapping function, the computer could provide a wealth of other information. Time of day, current speed, vessel speed, direction, rate of fuel consumption, estimated time to destination or check point, and other operating parameters could be displayed. Automatic radio tracking equipment could provide accurate positioning information. Depth information in coastal waters could be continuously updated and modified as a function of tide tables. The possibilities for this type of microcomputer application are almost limitless.

But for now, I guess I'll just keep drifting.

MICRO Letterbox

Dear Editor:

First, I don't know of any available printed material that has been as interesting and informative as "MICRO Magazine".

Many little problems related to programming have been removed due to the care and testing that is done, by writers, proofreaders, editors and by the production people. The quality is outstanding as witnessed by the brevity of your "Microbes" pages.

Now, the second part—I feel that I have received more value from this source than it has cost. Therefore, I offer this little tip for Apple II owners fortunate enough to have Disk II. Perhaps I am lazy, but my fingers get tired of constantly typing "catalog" following the end of a program. I know that many programs exist to change the disk command to cat or just plain 'C'. They are good, but why not insert these lines in place of "END" statement in the programs used frequently?

```
XX0  INPUT"WANT DISK
     CATALOG (Y/N) ?";A$
XX1  TEXT:HOME
XX2  IF A$ = "Y" THEN PRINT
     D$;"CATALOG"
```

Your program is still in memory should you choose to re-run it. Or with the catalog menu in front of you, a change to a different program is quite simple.

Another simple little tip is to type "VTAB < 1 THROUGH 20 >" to move the cursor up to the program desired, enter your command, and use the right arrow key to trace over the program listing, hit return, and your command is executed. Be sure that you don't leave any part of the program type or sector information before tracing over the program title. Those little left-overs produce nice error statements.

Thanks again for an excellent publication. I look forward to seeing it each month for it makes the Apple II more enjoyable for this retired telephone man who is pretty much housebound.

John A. Backman
302 North 76th
Seattle, WA 98103

Dear Editor:

I appreciated the letter by Robert V. Davis, MICRO, January 1981, but his letter didn't take full advantage of OSI's BASIC-in-ROM accuracy and he doesn't solve the absence of the PRINT USING command for anything but whole dollars.

That would be trouble if you're working in any accounting program where you need to keep track of pennies. The subroutine I am enclosing will print out amounts in dollars and cents from $0.00 to $167,772.15 with full accuracy and amounts close to one billion with 7+ decimal accuracy before going into scientific notation errors. Since Michigan income tax asks that you don't round off at one place, this program would keep you out of trouble with the taxman. Also by simply changing the value of H in line 20000 by a power of ten, and making the opposite change of T, you can set up for printing in the thousandth place or any other decimal place you wish with 7+ decimal accuracy. This routine will also increase the amount of decimals printed with any other BASIC computer.

```
50    INPUT"AMOUNT OF
      CASH";B: A=B
60    PRINT"BALANCE";:GOSUB
      20000
70    REM REST OF PROGRAM
19999 PRINTING SUBROUTINE
20000 H=100:T=1000:
      G=0:C=A:IF A>T*9
      THEN G=T*INT(A/T):C=
      A-G+T
20010 PLACE=INT(LOG
      (H)/LOG(10)+.5):IF A<1
      THEN 20070
20020 A$=STR$(INT(H* C+.5)):
      AC$= RIGHT$(A$,PL):
      B=LEN(A$)
20030 A$=LEFT$(A$,B-PL)+
      "."+AC$:IF G>0 THEN
      A$=STR$(G/T)+MID$
      (A$,3)
20040 PRINT TAB(20-LEN(A$))"$
      ";A$:RETURN
20060 REM AMOUNTS LESS
      THAN 1
20070 A$=STR$(INT(A*H
      +.5)/H):IF
      LEN(A$)<PL+2 THEN
      A$=A$+"0":
      GOTO20070
20080 GOTO20050
```

Dale Mayers
2301 S. Washington
Lansing, MI 48910

Dear Editor:

I'd like to share the following information in response to your Editorial in the January 1981 issue (MICRO Goes to School).

Our math department was given the job of learning how to operate the computers, then teach our students, then teach any interested non-math teachers. Granted, year #1 was trial and error. We spent many hours on our own time getting our act together.

Several members of the department formed a core group which learned how to program and joined area users groups, and then brought this information back to the rest of the department for general use. We subscribe to the leading magazines for help and greatly appreciate MICRO's help with the Club Circuit.

By using small ads, we have contacted and exchanged ideas, programs, and student booklets with teachers in several states. There is a vast network out there of independent math teachers which the computer will bring together.

This year, in our lab, we are more organized. Lab slots are assigned on a week-to-week basis and we have lab assignments sheets for the students, that they receive before they enter the lab. The sheets contain information as to what programs they should work on, what section of particular programs, what disks to use, which computers to be worked on, if the printer is to be used, etc. Thus, any computer center means preparation by the teachers involved if the center is to achieve its goals in the educational environment. And with the availability of data base programs, the department has its grades, orders, inventory, small supplies, etc., on disk.

Our computer center has taken a lot of effort, but it is well worth it. If any teacher or department requires more information, they may write to Apple Bit'N Pieces Educators Group c/o our school.

Patrick J. Calebrese
Math Dept. Chairman
Millcreek Township School District
Millcreek Middle School—J.S. Wilson
900 West 54th Street
Erie, PA 16509

S-C Assembler Modifications

The usefulness of the S-C assembler for the Apple can be enhanced with the addition of a command to automatically generate line numbers for the programmer while he is entering the source code.

Ned W. Rhodes
2001 No. Kenilworth St.
Arlington, Virginia 22205

The S-C assembler is one of the many assemblers available for the Apple computer system. The original version of the S-C assembler was cassette-based and performed well for the user with a minimal system. Subsequent versions of the assembler have been disk-based. With the announcement of version 3.2, previous owners were invited to upgrade their assemblers for $12.50. This I did, and along with my upgrade kit came information on how the S-C assembler could be modified to incorporate more features. In this article I will describe modifications to the S-C assembler that allow the S-C assembler to work with the auto-start ROM, automatically generate line numbers for source code entry, and allow the user to change the starting line number and increment for the auto-line numbering mode.

Adding Back the Multiply Routine

In the auto-start ROM, Apple has incorporated some features that make line editing easier and allow the Apple to automatically boot itself when power is applied. In order to give us all of these features, they had to replace some old (and very useful) code in the F8 ROM with their new routines. One of the deleted routines happened to be the Integer multiply routine which is used by the S-C assembler. So, if you have the auto-start ROM, you must patch the assembler and add the multiply code in order to make the

Listing 1

```
                  1000 *
                  1010 *
                  1020 *
                  1030 *
                  1040 *
                  1050 *
                  1060 *
                  1070 *
                  1080 *     ROUTINE TO DO AUTO NUMBERING
                  1090 *
                  1100 *
                  1110       .OR $1D21      AFTER MULTIPLY ROUTINE
                  1120 *
                  1130 *
                  1140 *     DO WE DO IT??
                  1150 *
                  1160 *
1D21- 2C 7D 1D    1170 ANUM BIT FLAG      TEST AUTO-FLAG
1D24- 10 27       1180      BPL EXIT      NOT TURNED ON
1D26- 4E 7D 1D    1190      LSR FLAG      CLEAR THE FLAG
1D29- E0 00       1200      CPX #0        IF NOT IN COLUMN 1
1D2B- D0 20       1210      BNE EXIT      THEN DON'T DO IT
                  1220 *
                  1230 *
                  1240 *     PRINT IT OUT, AND STORE IN
                  1250 *     INPUT BUFFER
                  1260 *
                  1270 *
1D2D- 20 5C 1D    1280      JSR CONV4     CONVERT LINE NUMBER
                  1290 *
                  1300 *
                  1310 *     INCREMENT THE LINE NUMBER
                  1320 *
                  1330 *
1D30- F8          1340      SED           SET DECIMAL MODE
1D31- 18          1350      CLC           CLEAR CARRY
1D32- AD 7A 1D    1360      LDA NUM+1     ADD LSB'S
1D35- 6D 7C 1D    1370      ADC INC+1
1D38- 8D 7A 1D    1380      STA NUM+1
1D3B- AD 79 1D    1390      LDA NUM       ADD MSB'S
1D3E- 6D 7B 1D    1400      ADC INC
1D41- 8D 79 1D    1410      STA NUM
1D44- D8          1420      CLD           SET BINARY MODE
1D45- A9 A0       1430      LDA #$A0      SEND A SPACE TOO
1D47- 20 72 1D    1440      JSR CHO
1D4A- 4C 0C FD    1450      JMP $FD0C     INPUT NEXT CHARACTER
                  1460 *
                  1470 *
                  1480 *     CALL THE MONITOR TO READ KEY
                  1490 *     CHECK IF CONTROL-N.
                  1500 *     IF SO, SET AUTO-FLAG AND
                  1510 *     CHANGE TO CARRIAGE RETURN
                  1520 *
                  1530 *
1D4D- 20 1B FD    1540 EXIT JSR $FD1B     MONITOR KEYIN
1D50- C9 8E       1550      CMP #$8E      CONTROL-N??
1D52- D0 05       1560      BNE RTRN      NO
1D54- A9 8D       1570      LDA #$8D      CHANGE TO CONTROL-M
1D56- 8D 7D 1D    1580      STA FLAG      SET FLAG
1D59- 4C 8B 13    1590 RTRN JMP $138B     RE-JOIN SCAII
                  1600 *
                  1610 *
                  1620 *     CONVERT AND STORE FOUR DIGITS
                  1630 *
                  1640 *
                  1650 CONV4
1D5C- AD 79 1D    1660      LDA NUM       FIRST TWO DIGITS
1D5F- 20 65 1D    1670      JSR CONV2
1D62- AD 7A 1D    1680      LDA NUM+1     LAST TWO DIGITS
                  1690 *
                  1700 *
                  1710 *     CONVERT AND STORE TWO DIGITS
                  1720 *
```

assembler run properly. Bob Sander-Cederlof (the S-C assembler creator) included the patch along with my upgrade kit and I will repeat it here.

Before we can patch the assembler, we have to create some room for the patch. Bob suggested that we move the starting address of the symbol table up a page or two, and make all patches and modifications in this new space. The assembler resides in memory from $1000 through $1BFF, and the symbol table follows, starting at $1C00. The moving of the symbol table is accomplished by changing location $1010 in the assembler. Now, I suggest that we start the symbol table at $1E00 so that we have plenty of room for the enhancements that are to be described later on. The step-by-step instructions for moving the symbol table are:

1. Load the assembler
2. Change contents of $1010 to $1E
3. Re-save the assembler using BSAVE ASMB,A$1000,L$E00

Note that the older versions of the assembler may also be patched in this fashion, but that the address to be patched will not necessarily be the same. In that case, use the Monitor disassembler and examine memory on either side of address $1010 until you find either a $1C or $1D, as that was the default-starting page number of the symbol table.

The multiple routine may now be added, starting at location $1D00 using the monitor insert command.

```
*1D00: A0 10 A5 50 4A 90 0C 18
       A2 FE B5 54 75 56 95 54
*1D10: E8 D0 F7 A2 03 76 50 CA
       10 FB 88 D0 E5 60
```

And finally, we need to change the JSR instruction that points to the multiply routine to point to the relocated code for the multiply routine. You should find a JSR $FB63 at location $1122. The following will change the destination address to $1D00.

```
*1123: 00 1D
```

Now, the assembler may be saved as instructed in step 3 above. This modified version of the assembler will now work properly with the auto-start ROM.

Automatic Line Numbers

The other little goodie that Bob included in my upgrade kit was a routine that allowed the assembler to automatically generate line numbers so

```
                 1730 *
                 1740 CONV2
1D65- 48         1750        PHA           SAVE BYTE ON STACK
1D66- 4A         1760        LSR
1D67- 4A         1770        LSR           GET LEFT DIGIT
1D68- 4A         1780        LSR
1D69- 4A         1790        LSR
1D6A- 20 70 1D   1800        JSR CONV1     CONVERT AND STORE IT
1D6D- 68         1810        PLA           GET BYTE FROM STACK
1D6E- 29 0F      1820        AND #$0F      ISOLATE SECOND DIGIT
                 1830 CONV1
1D70- 09 B0      1840        ORA #$B0      CONVERT TO ASCII
1D72- 9D 00 02   1850 CHO    STA $200,X    STORE IN INPUT BUFFER
1D75- E8         1860        INX           INCREMENT BUFFER POINTER
1D76- 4C ED FD   1870        JMP $FDED     PRINT THE CHARACTER
                 1880 *
1D79- 10 00      1890 NUM   .HS 1000       INITIAL NUMBER
1D7B- 00 10      1900 INC   .HS 0010       INCREMENT
1D7D- 00         1910 FLAG  .HS 00
                 1920       .END

SYMBOL TABLE

ANUM   1D21   EXIT   1D4D   RTRN   1D59
CONV4  1D5C   CONV2  1D65   CONV1  1D70
CHO    1D72   NUM    1D79   INC    1D7B
FLAG   1D7D
```

Listing 2

```
                 1000 *
                 1010 *
                 1020 *
                 1030 *
                 1040 *
                 1050 *
                 1060 *
                 1070 *     THIS ADDS THE AUTO COMMAND TO THE
                 1080 *     S-C ASSEMBLER.  THE AUTO COMMAND
                 1090 *     ALLOWS YOU TO SET THE STARTING LINE
                 1100 *     NUMBER AND THE INCREMENT FOR AUTOMATIC
                 1110 *     LINE ENTRY.
                 1120 *
                 1130 *     THE FORMAT OF THE COMMAND WILL BE:
                 1140 *
                 1150 *     AUTO START,INC
                 1160 *
                 1170 *
                 1180 SPACE .EQ $20       SPACE
                 1190 LBUF .EQ $200       LINE BUFFER
                 1200 COMMA .EQ $2C       COMMA
                 1210 WARM .EQ $1003      WARM START
                 1220 NUM  .EQ $1D79      STARTING LINE NUMBER
                 1230 INC  .EQ $1D7B      INCREMENT
                 1240      .OR $1D7E      AFTER THE AUTO LINE NUMBER GENERATOR
                 1250 *
                 1260 *
                 1270 *
1D7E- A0 03      1280 AUTO LDY #3         START AT FOURTHCHARACTER
1D80- B9 00 02   1290 SLOP LDA LBUF,Y     GET CHARACTER
1D83- F0 60      1300      BEQ DONE       ALL DONE--DO NOTHING
1D85- C9 20      1310      CMP #SPACE     IS IT A SPACE??
1D87- F0 03      1320      BEQ GSPAC      YES
1D89- C8         1330      INY            BUMP Y
1D8A- D0 F4      1340      BNE SLOP       IDLE UNTIL A SPACE
                 1350 *
                 1360 *
                 1370 *     GOT A SPACE.  IDLE UNTIL NO MORE SPACES
                 1380 *
                 1390 *
                 1400 GSPAC
1D8C- C8         1410      INY            BUMP Y
1D8D- B9 00 02   1420      LDA LBUF,Y     GET CHARACTER
1D90- F0 53      1430      BEQ DONE       WE ARE DONE
1D92- C9 20      1440      CMP #SPACE     IS IT A SPACE
1D94- F0 F6      1450      BEQ GSPAC      LOOP UNTIL NO SPACE
                 1460 *
                 1470 *
                 1480 *     COUNT THE NUMBER OF CHARACTERS
                 1490 *     UNTIL THE "," AND SAVE THE POSITION
                 1500 *     NUMBER OF THE LAST CHARACTER.
                 1510 *
                 1520 *
1D96- A2 00      1530      LDX #00        GET A ZERO
1D98- C9 2C      1540 CLOP CMP #COMMA     IS IT A COMMA??
1D9A- F0 09      1550      BEQ SAVIT      YES
1D9C- E8         1560      INX            BUMP COUNT
1D9D- C8         1570      INY            BUMP CHARACTER SCAN
1D9E- B9 00 02   1580      LDA LBUF,Y     GET NEXT CHARACTER
1DA1- F0 02      1590      BEQ SAVIT      SAVE PARAMETERS
1DA3- D0 F3      1600      BNE CLOP       TRY AGAIN
                 1610 *
                 1620 *
                 1630 *     WE GET HERE AND SAVE X AND Y FOR LATER
```

```
                 *
                 *
                 SAVIT
1DA5- 8E 2D 1E          STX SCNT      SAVE COUNT
1DA8- 8C 2E 1E          STY EPOS      END POSITION + 1
1DAB- B9 00 02          LDA LBUF,Y    GET CHARACTER AGAIN
1DAE- F0 1E             BEQ DSTRT     IF ZERO GO AWAY
                 *
                 *
                 *      SCAN THE INCREMENT
                 *
                 *
1DB0- C8                INY           NEXT CHARACTER
1DB1- A2 00             LDX #00       ZERO COUNT
1DB3- B9 00 02   ILOP   LDA LBUF,Y    GET CHARACTER
1DB6- F0 04             BEQ DINC      DONE WITH SCAN
1DB8- E8                INX           BUMP COUNT
1DB9- C8                INY           NEXT CHARACTER
1DBA- D0 F7             BNE ILOP      REPEAT TIL DONE
                 *
                 *
                 *      CONVERT THE INCREMENT AND SAVE
                 *
                 *
1DBC- 8A         DINC   TXA           SET CONDITION CODE
1DBD- F0 0F             BEQ DSTRT     IF ZERO DO START
1DBF- 20 E8 1D          JSR GETNUM    CONVERT NUMBER
1DC2- AD 2F 1E          LDA HOLD      GET MSB
1DC5- 8D 7B 1D          STA INC       SAVE
1DC8- AD 30 1E          LDA HOLD+1    GET LSB
1DCB- 8D 7C 1D          STA INC+1     SAVE
                 *
                 *
                 *      DO THE START LINE NUMBER
                 *
                 *
                 DSTRT
1DCE- AE 2D 1E          LDX SCNT      GET COUNT
1DD1- F0 12             BEQ DONE      IF ZERO -- IGNORE
1DD3- AC 2E 1E          LDY EPOS      GET POSITION
1DD6- 20 E8 1D          JSR GETNUM    CONVERT
1DD9- AD 2F 1E          LDA HOLD      GET MSB
1DDC- 8D 79 1D          STA NUM       SAVE
1DDF- AD 30 1E          LDA HOLD+1    GET LSB
1DE2- 8D 7A 1D          STA NUM+1     SAVE
                 *
                 *
                 *      DONE OR ABORT
                 *
                 *
1DE5- 4C 03 10   DONE   JMP WARM      WARM START
                 *
                 *
                 *      GETNUM -- CONVERTS ASCII TO BCD
                 *
                 *
                 GETNUM
1DE8- A9 00             LDA #00       GET A ZERO
1DEA- 8D 2F 1E          STA HOLD      ZERO OUT
1DED- 8D 30 1E          STA HOLD+1    ZERO OUT
1DF0- 20 20 1E          JSR READ8     GET 8 BITS
1DF3- 8D 30 1E          STA HOLD+1    SAVE BITS
1DF6- CA                DEX           DECREMENT LOOP COUNT
1DF7- F0 26             BEQ EXT       DONE
1DF9- 20 20 1E          JSR READ8     GET 8 BITS
1DFC- 20 27 1E          JSR SHIFT     SHIFT LEFT 4

1DFF- 18                CLC           CLEAR CARRY
1E00- 6D 30 1E          ADC HOLD+1    PUT IN BITS
1E03- 8D 30 1E          STA HOLD+1    SAVE BACK
1E06- CA                DEX           DECREMENT LOOP COUNT
1E07- F0 16             BEQ EXT       DONE
1E09- 20 20 1E          JSR READ8     GET 8 BITS
1E0C- 8D 2F 1E          STA HOLD      SAVE BITS
1E0F- CA                DEX           DECREMENT LOOP COUNT
1E10- F0 0D             BEQ EXT       DONE
1E12- 20 20 1E          JSR READ8     GET BITS
1E15- 20 27 1E          JSR SHIFT     SHIFT LEFT 4
1E18- 18                CLC           CLEAR CARRY
1E19- 6D 2F 1E          ADC HOLD      ADD IN BITS
1E1C- 8D 2F 1E          STA HOLD      SAVE BACK
1E1F- 60         EXT    RTS           RETURN
                 *
                 *
                 *      READ8 -- READ 8 BITS FROM LINE BUFFER
                 *
                 *
                 READ8
1E20- 88                DEY           DECREASE POINTER
1E21- B9 00 02          LDA LBUF,Y    GET CHARACTER
1E24- 29 0F             AND #$0F      ONLY FOUR BITS
```

as to relieve the programmer of that task. I have often wanted that sort of a feature when I am doing a lot of coding with the S-C assembler. I have included the code that will automatically generate the line numbers in listing 1. It is placed immediately after the multiply routine that is listed in the previous section. The steps that are required to incorporate the routine into the assembler are:

1. BRUN the asssembler
2. Enter the source code from listing 1
3. Assemble the code using the assembler
4. Patch an assembler address that will allow access to the auto-line routine. Location $1388 should contain a JSR $FD1B. Change the address to $1D21 using the monitor command:

   *1389:21 1D.

The automatic line number routine is started by typing a control-N instead of a RETURN. So, whenever you type control-N, the assembler will generate a carriage return, a line feed, and then display the next line number on the screen. I incorporated this routine in my assembler and was very happy with it with one exception. In order to change either the starting line number or the increment, you had to change the values stored in memory. This soon got to be very tedious, especially when I had to refer to the source listing in order to find the address that I had to change if I needed a different starting line number or increment. I longed for a command to change one or both of the numbers.

The 'AUTO' Command for the S-C Assembler

Listing 2 is the code to include the 'AUTO' command to the S-C assembler. The format of the AUTO command is the same as for Integer BASIC, which is:

AUTO starting line number, increment.

The design of the routine is quite simple. First the routine goes to the input line buffer and begins to scan the command, beginning with character four. It throws away all characters until it finds a space. This is done so that the user may type any character string that starts with the first three letters 'AUT'. After we have encountered a space, we count the number of characters from there until the comma. This is the number of digits in the starting line number and this value is saved for later use. Note that this value can be zero,

which implies that you can change only the increment, but don't have to change the starting line number also (for example AUTO ,10).

Next we scan the character string, starting with the first character after the comma, and ending with the null byte that terminates the input buffer string. Again the number of characters is saved and, as mentioned above, it also may be zero if you only want to change the starting line number and keep the same increment (for example AUTO 1000). The increment character string is converted from ASCII to BCD by the GETNUM routine. The resulting BCD number for the increment is saved as the new increment. Finally, the starting line number string is converted to BCD and saved as the new starting line number. Then we jump back to the assembler command mode.

Only a small problem now exists—there is no 'AUTO' command in the basic S-C assembler. We have two options: we can find the command dispatch table in the assembler and add another command to it (this may be complicated), or we can replace one of the existing commands with our new command. I chose to do the latter. The code at the end of listing 2 changes the 'JOIN' command to 'AUTO' by changing the ASCII command string and the address of the routine that actually does the command in the command dispatch table. As before, the code needs to be assembled as part of the assembler and saved as indicated above.

I have recommended that you create a source file and assemble that in order to incorporate these new features. This is not necessary, since I have included the object code as part of the listings. Instead, you could just enter the object code directly into memory and make the patches listed above. The only problem that I see with that method, is that it can be very tedious, if you were to make a small mistake. Also, it is a good idea to make yourself a back-up copy of the assembler until you have tested out your new and improved version.

Ned W. Rhodes received his BSEE from the University of Minnesota and his MS in Computer Science from the George Washington University in Washington D.C. He is currently employed by the David W. Taylor Naval Ship Research and Development Center, where he develops high-speed minicomputer-based data acquisition systems for use during full-scale trials aboard naval vessels.

MICRO

```
1E26- 60        2550      RTS          DONE
                2560 *
                2570 *
                2580
                2590 *    SHIFT -- SHIFT LEFT 4 BITS
                2600 *
                2610 *
                2620 SHIFT
1E27- 18        2630      CLC          CLEAR CARRY
1E28- 2A        2640      ROL          LEFT 1
1E29- 2A        2650      ROL          LEFT 1
1E2A- 2A        2660      ROL          LEFT 1
1E2B- 2A        2670      ROL          LEFT 1
1E2C- 60        2680      RTS          RETURN
                2690 *
                2700 *
                2710 *    STORAGE
                2720 *
                2730 *
1E2D- 00        2740 SCNT .HS 00       COUNT
1E2E- 00        2750 EPOS .HS 00       POSITION
1E2F- 00 00     2760 HOLD .DA 0        TEMP STORAGE
                2770 *
                2780 *
                2790 *    CHANGE JOIN TO AUTO
                2800 *
                2810 *
                2820      .OR $1289    IN DISPATCH TABLE
1289- 41 55 54  2830      .AS /AUT/    AUTO COMMAND
128C- 7E 1D     2840      .DA AUTO     ADDRESS
                2850      .EN
```

PRINT USING for the PET

This is an excellent PET version of Gary Morris' Apple program. It runs on any PET or CBM machine.

David Malmberg
43064 Via Moraga
Fremont, California 94538

Gary Morris recently published a PRINT USING program for the Apple in the October 1980 issue of *MICRO* (29:14). His program made use of some of the ROM routines in Applesoft. Since Microsoft developed both Applesoft and the various versions of PET BASIC, I felt that Gary's routine could be easily adapted for the PET. After consulting Jim Butterfield's many memory maps of the PET ROMs, and a fair amount of experimentation, I succeeded in modifying Gary's basic routine to work on the PET.

Listing 1 is a BASIC program that POKEs the machine code for the routine into the second cassette buffer (from 826 to 984). This program also detects which of the three versions of BASIC is operable in the specific PET and modifies the machine code accordingly. This is done by PEEKing into location 50003 which contains a "0" for BASIC 2.0, a "1" for BASIC 3.0, and a "160" for BASIC 4.0. The program in listing 1 will also set the USR vector (locations 1 and 2) to point to the beginning of the code in the second cassette buffer. Once this program has been run, the machine language routine is available to any BASIC program via the USR function.

As an example of how this would be used, consider the following BASIC instructions:

```
10 ED$ = "$ , 0.00"
20 X = 123456
30 PRINT"TOTAL IS ";:Y =
   USR(X)
```

This sequence will cause the following to be printed:

```
TOTAL IS $ 1,234.56
```

The edit pattern to be used in formatting the output must be specified by the string variable ED$. The edit pattern may contain almost any valid character (such as, $ #, %, ' = / K . etc.). These characters will be "skipped over" and the various digits of the number will be inserted into the blanks of the edit pattern, or overlaid on any 0's in the pattern. The value to be printed will be edited from right to left. If the value is too large for the edit field, the left-most characters will be truncated. A comma in the pattern will be printed only if there is at least one digit to the left of it. If the value is negative, the minus sign will be placed to the left of the highest digit.

The value to be edited is passed to the PRINT USING routine as the parameter of the USR function, e.g., X in the previous example. This parameter may be a complex expression, rather than just a variable or a numeric value. The value returned by the USR function to BASIC (Y in the previous example) will be garbage and have no meaning. Be careful not to use a variable name that is significant to the rest of the program as the left hand side of the USR equation.

Listing 1

```
 10 PRINT "[CLR][ 2 DOWN]PRINT USING FOR THE PET"
 20 PRINT "[DOWN]BY DAVID MALMBERG"
 30 REM ADAPTED FROM A ROUTINE FOR THE APPLE
 40 REM BY GARY MORRIS IN OCT-1980 MICRO
 50 PRINT "[HOME]           LOADING 2ND CASSETTE BUFFER"
 60 FOR I= 826 TO 984 :READ DC:POKE I,DC:PRINT "[HOME]";I;DC:NEXT I
 70 REM BASIC 3.0 VERSION
 80 DATA 32,233,220,169,69,162,196,133
 90 DATA 66,134,67,32,201,207,160,2,177
100 DATA 68,133,93,136,177,68,133,92,136
110 DATA 177,68,201,16,144,2,169,16,141
120 DATA 32,3,168,136,177,92,153,33,3
130 DATA 136,16,248,160,0,185,0,1,240
140 DATA 3,200,208,248,174,32,3,136,185
150 DATA 0,1,72,104,72,201,45,208,14,189
160 DATA 32,3,201,45,144,22,202,208,240
170 DATA 104,24,144,54,189,32,3,201,32
180 DATA 240,8,201,44,240,238,201,48,144
190 DATA 234,104,157,32,3,202,240,32,192
200 DATA 1,208,205,232,24,144,16,189,32
210 DATA 3,201,36,240,17,201,46,176,5
220 DATA 169,32,157,32,3,202,240,5,236
230 DATA 32,3,144,232,174,32,3,169,0,157
240 DATA 33,3,160,3,169,33,32,28,202,169
250 DATA 0,133,7,96
260 POKE 1,58:POKE 2,3:REM SET USR VECTOR
270 IF PEEK(50003)<>160 THEN 310
280 REM BASIC 4.0 MODIFICATIONS
290 POKE 827,147:POKE 828,207:POKE 838,135:POKE 839,193
300 POKE 978,29:POKE 979,187
310 IF PEEK(50003)<>0 THEN 370
320 REM BASIC 2.0 MODIFICATIONS
330 POKE 827,175:POKE 828,220:POKE 838,215:POKE 839,207
340 POKE 978,39:POKE 979,202:POKE 983,94
350 POKE 834,148:POKE 836,149:POKE 843,150:POKE 845,175
360 POKE 848,150:POKE 850,174:POKE 853,150:POKE 866,174
370 PRINT "[ 6 DOWN]LOADING COMPLETED"
```

The routine works by editing the ASCII representation of the number passed as the USR parameter. The routine assumes that this value has been "integerized" and that the ASCII representation does not contain a decimal point. The position of the decimal point (if any) will be implied by the edit pattern, i.e., the variable ED$.

If the actual value you wish to format has a decimal point, or if you wish to scale the number to be printed differently from the way it is represented internally in the PET, you can use a BASIC user-defined function to handle the conversion before going to the USR routine. For example:

```
10 DEF FNS2(X) = INT (X*100
     + 0.5)
20 DEF FNPK(X) = INT(X/2.21
     + 0.5)
30 ED$ = "$   ,     0.00"
40 Y = USR(FNS2(12.3456))
50 ED$ = "KILOS=        "
60 PRINT
70 Z = 1000.0 : REM POUNDS
80 Y = USR(FNPK(Z))
```

will cause the following output:

```
$ 12.35
KILOS= 452
```

Listing 2 gives the assembler source code for the PET PRINT USING routine. The appropriate ROM routine locations are given for all three versions of PET BASIC, with conditional assembly determined by the value of ROMs in line 100. The assembled code shown along side of the source code is for BASIC 3.0—the "new" ROMs. The assembler source is almost identical to that shown in Gary Morris' original Applesoft article, with the exception of the use of the STROUT ROM routine to print the formatted representation of the number (line 1450). The assembler source also has several slight differences to accommodate the differences between how Applesoft and PET BASICs handle the ASCII representation of numbers, and the value the USR function returns. The assembler source is well-commented and is very straightforward.

Listing 2

```
                 0010 ;PRINT USING FOR THE PET
                 0020 ;BY DAVID MALMBERG
                 0030 ;43064 VIA MORAGA
                 0040 ;FREMONT, CALIFORNIA 94538
                 0050 ;
                 0060 ;ADAPTED FROM A ROUTINE FOR THE APPLE
                 0070 ;BY GARY MORRIS IN OCT-1980 MICRO
                 0080 ;
                 0090            .BA $033A
                 0100 ROMS       .DE 3
                 0110            .OS
                 0120 STRING     .DE $100
                 0130 LENGTH     .DE 800
                 0140 EDITBUF    .DE 801
                 0150 ;
                 0160            IFE ROMS-3
                 0170 FLPASC     .DE $DCE9
                 0180 STROUT     .DE $CA1C
                 0190 NAME       .DE $42
                 0200 VARIABLE   .DE $44
                 0210 FIND       .DE $CFC9
                 0220 PNTR       .DE $5C
                 0230 VARTYP     .DE $07
                 0240            ***
                 0250 ;
                 0260            IFE ROMS-4
                 0270 FLPASC     .DE $CF93
                 0280 STROUT     .DE $BB1D
                 0290 NAME       .DE $42
                 0300 VARIABLE   .DE $44
                 0310 FIND       .DE $C187
                 0320 PNTR       .DE $5C
                 0330 VARTYP     .DE $07
                 0340            ***
                 0350 ;
                 0360            IFE ROMS-2
                 0370 FLPASC     .DE $DCAF
                 0380 STROUT     .DE $CA27
                 0390 NAME       .DE $94
                 0400 VARIABLE   .DE $96
                 0410 FIND       .DE $CFD7
                 0420 PNTR       .DE $AE
                 0430 VARTYP     .DE $5E
                 0440            ***
                 0450 ;
                 0460 ;FIRST CONVERT NUMBER PASSED BY USR
                 0470 ;FUNCTION TO FLOATING ACCUM TO
                 0480 ;ASCII STRING STARTING AT 'STRING'
                 0490 ;
033A- 20 E9 DC   0500            JSR FLPASC
                 0510 ;NOW FIND THE VARIABLE (ED$) TO USE
                 0520 ;IN THE EDIT PATTERN
                 0530 ;
033D- A9 45      0540 SEARCH     LDA #'E        ;BASIC VARIABLE
033F- A2 C4      0550            LDX #$C4       ;NAME IS ED$
0341- 85 42      0560            STA *NAME
0343- 86 43      0570            STX *NAME+1
0345- 20 C9 CF   0580            JSR FIND
0348- A0 02      0590            LDY #2
034A- B1 44      0600            LDA (VARIABLE),Y     ;GET ADDR HI
034C- 85 5D      0610            STA *PNTR+1
034E- 88         0620            DEY
034F- B1 44      0630            LDA (VARIABLE),Y     ;GET ADDR LO
0351- 85 5C      0640            STA *PNTR
0353- 88         0650            DEY
0354- B1 44      0660            LDA (VARIABLE),Y     ;GET LENGTH
0356- C9 10      0670            CMP #16
0358- 90 02      0680            BCC LENOK      ;MAXIMUM LENGTH
035A- A9 10      0690            LDA #16        ;ALLOWED IS 16!!!
035C- 8D 20 03   0700 LENOK      STA LENGTH
                 0710 ;
                 0720 ;MOVE THE ED$ PATTERN TO EDITBUF
035F- A8         0730            TAY
0360- 88         0740            DEY
0361- B1 5C      0750 LOOP2      LDA (PNTR),Y
0363- 99 21 03   0760            STA EDITBUF,Y
0366- 88         0770            DEY
0367- 10 F8      0780            BPL LOOP2
                 0790 ;
                 0800 ;FIND THE ASCII STRING END
0369- A0 00      0810            LDY #0
036B- B9 00 01   0820 LOOP       LDA STRING,Y          ;GET CHAR
```

```
036E- F0 03     0830            BEQ EDIT
0370- C8        0840            INY
0371- D0 F8     0850            BNE LOOP
                0860 ;MOVE STRING TO THE EDITBUF, FROM RIGHT
                0870 ;TO LEFT, FILLING OVER NUMBERS BUT
                0880 ;SKIPPING COMMA'S AND PERIODS.
                0890 ; IF WE COME TO A MINUS SIGN THEN
                0900 ;KEEP GOING LEFT UNTIL THE PATTERN
                0910 ;HAS A BLANK OR A COMMA, THEN KEEP
                0920 ;GOING LEFT STORING BLANKS IN THE
                0930 ;EDITBUF UNTIL IT ENDS OR WE COME
                0940 ;TO A DOLLAR SIGN
                0950 ;
0373- AE 20 03  0960 EDIT       LDX LENGTH    ;FIELD WIDTH
                0970 ;
0376- 88        0980 EDLOOP     DEY
0377- B9 00 01  0990            LDA STRING,Y          ;GET CHARACTER
037A- 48        1000            PHA   ;SAVE IT
037B- 68        1010 CHECK      PLA
037C- 48        1020            PHA
037D- C9 2D     1030            CMP #'-        ;IF A MINUS SIGN
037F- D0 0E     1040            BNE DIGIT     ;SKIP TO A BLANK
0381- BD 20 03  1050 MINUS      LDA EDITBUF-1,X
0384- C9 2D     1060            CMP #'-
0386- 90 16     1070            BCC DROPIT
0388- CA        1080 SKIPIT     DEX
0389- D0 F0     1090            BNE CHECK
038B- 68        1100            PLA
038C- 18        1110            CLC
038D- 90 36     1120            BCC DONE
038F- BD 20 03  1130 DIGIT      LDA EDITBUF-1,X
0392- C9 20     1140            CMP #'
0394- F0 08     1150            BEQ DROPIT
0396- C9 2C     1160            CMP #',
0398- F0 EE     1170            BEQ SKIPIT
039A- C9 30     1180            CMP #'0
039C- 90 EA     1190            BCC SKIPIT
039E- 68        1200 DROPIT     PLA   ;GET IT BACK
039F- 9D 20 03  1210            STA EDITBUF-1,X
03A2- CA        1220            DEX
03A3- F0 20     1230            BEQ DONE
03A5- C0 01     1240            CPY #1        ;END OF STRING?
03A7- D0 CD     1250            BNE EDLOOP
03A9- E8        1260            INX
03AA- 18        1270            CLC
03AB- 90 10     1280            BCC NEXT1
03AD- BD 20 03  1290 BLANK      LDA EDITBUF-1,X       ;BLANK FROM
03B0- C9 24     1300            CMP #'$       ;HERE TO $
03B2- F0 11     1310            BEQ DONE
03B4- C9 2E     1320            CMP #'.
03B6- B0 05     1330            BCS NEXT1
03B8- A9 20     1340            LDA #'
03BA- 9D 20 03  1350            STA EDITBUF-1,X
03BD- CA        1360 NEXT1      DEX
03BE- F0 05     1370            BEQ DONE
03C0- EC 20 03  1380            CPX LENGTH
03C3- 90 E8     1390            BCC BLANK
03C5- AE 20 03  1400 DONE       LDX LENGTH
03C8- A9 00     1410            LDA #0
03CA- 9D 21 03  1420            STA EDITBUF,X         ;PUT 0 AT END
03CD- A0 03     1430            LDY #H, EDITBUF
03CF- A9 21     1440            LDA #L, EDITBUF
03D1- 20 1C CA  1450            JSR STROUT
03D4- A9 00     1460            LDA #0        ;SET TYPE TO NUMERIC
03D6- 85 07     1470            STA *VARTYP   ;TO AVOID BASIC ERROR
03D8- 60        1480            RTS
                1490 XEND       .EN

LABEL FILE:  [ / = EXTERNAL ]

/ROMS=0003              /STRING=0100            /LENGTH=0320
/EDITBUF=0321           /FLPASC=DCE9            /STROUT=CA1C
/NAME=0042              /VARIABLE=0044          /FIND=CFC9
/PNTR=005C              /VARTYP=0007            SEARCH=033D
LENOK=035C              LOOP2=0361              LOOP=036B
EDIT=0373               EDLOOP=0376             CHECK=037B
MINUS=0381              SKIPIT=0388             DIGIT=038F
DROPIT=039E             BLANK=03AD              NEXT1=03BD
DONE=03C5               XEND=03D9
```

MICRO

MICRO Club Circuit

Mike Rowe
Club Circuit
P.O. Box 6502
Chelmsford, MA 01824

The following club announcements are presented in zip code order.

Richmond Computer Club
Gary F. Cowardin is Treasurer for this group which meets on the last Monday of each month at 7:30 pm at the Science Museum of VA. This club has a membership of over 50 active members who meet to encourage organized computer use involving Ohio Scientific, Heath, TRS-80, Apple, and many other microcomputers. For further information, write:

Secretary
1004 Lorraine Avenue
Richmond, VA 23227

Jacksonville Atari & PET Society (JAPS)
This group meets at various member's homes and businesses to assist members, exchange ideas, information and experiences. Russell A. Grokett, Jr is president for this newly-formed group. For monthly information on club meeting locations contact the president at (904) 725-0435 evenings and weekends. Or write to:

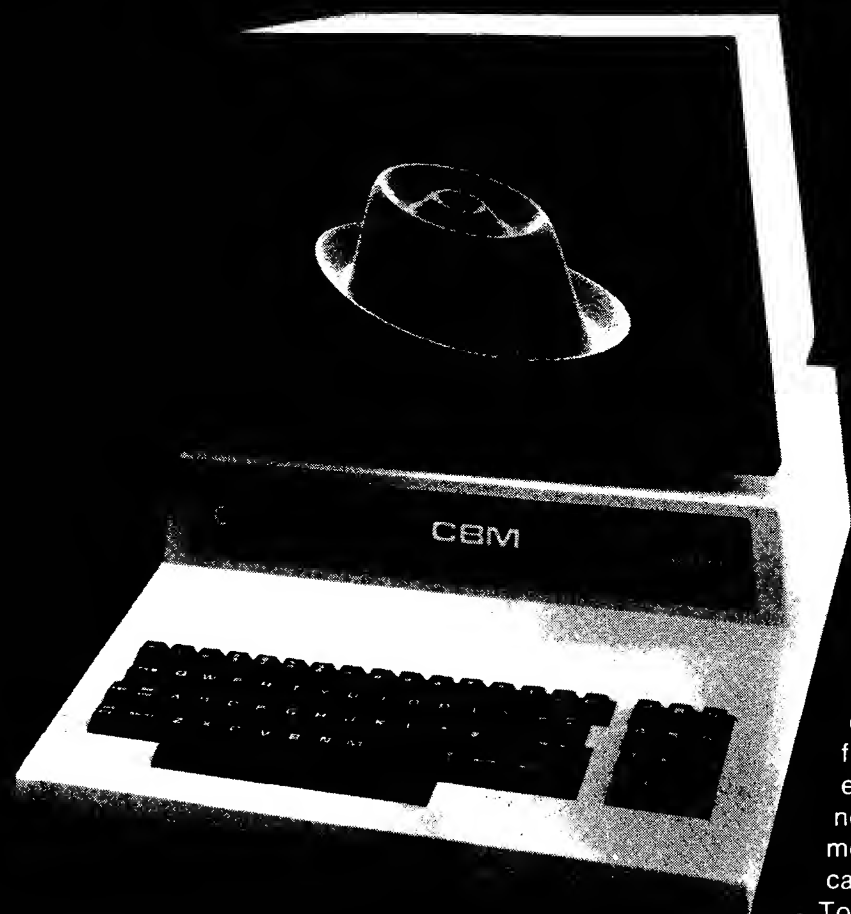
401 Monument Road #171
Jacksonville, FL 32211

6502 User's Group
Chairman Gerald Key heads this group of 28+ members which meets every 3rd Thursday of the month at 7:30 pm. Meetings are held at the State Savings Bank Community Room, 444 Havens Corner Road, Gahanna, OH. This club states its purpose as a means to exchange ideas, provide assistance to members, and promote the use of microcomputers. This club provides a forum for all 6502-based users and is the only Columbus area alternative to many Apple user's groups. For further information, write:

Chairman
141 Flintridge Drive
Gahanna, OH 43230

(Continued on page 20)

80 COLUMN GRAPHICS



The Integrated Visible Memory for the PET has now been redesigned for the new 12" screen 80 column and forthcoming 40 column PET computers from Commodore. Like earlier MTU units, the new K-1008-43 package mounts inside the PET case for total protection. To make the power and flexibility of the 320 by 200 bit mapped pixel graphics display easily accessible, we have designed the Keyword Graphic Program. This adds 45 graphics commands to Commodore BASIC. If you have been waiting for easy to use, high resolution graphics for your PET, isn't it time you called MTU?

K-1008-43M Manual only $10 (credited toward purchase)
k-1008-43 Complete ready to install package $495

Mastercharge and Visa accepted

Write or call today for our full line catalog describing all MTU 6502 products, including our high speed 8" Floppy Disk Controller for up to 4 megabytes of PET storage.

The image on the screen was created by the program below.

```
 10 VISMEM: CLEAR
 20 P=160: Q=100
 30 XP=144: XR=1.5*3.1415927
 40 YP=55: YR=1: ZP=64
 50 XF=XR/XP: YF=YP/YR: ZF=XR/ZP
 60 FOR ZI=-Q TO Q-1
 70 IF ZI<-ZP OR ZI>ZP GOTO 150
 80 ZT=ZI*XP/ZP: ZZ=ZI
 90 XL=INT(.5+SQR(XP*XP-ZT*ZT))
100 FOR XI=-XL TO XL
110 XT=SQR(XI*XI+ZT*ZT)*XF: XX=XI
120 YY=(SIN(XT)+.4*SIN(3*XT))*YF
130 GOSUB 170
140 NEXT XI
150 NEXT ZI
160 STOP
170 X1=XX+ZZ+P
180 Y1=YY-ZZ+Q
190 GMODE 1: MOVE X1,Y1: WRPIX
200 IF Y1=0 GOTO 220
210 GMODE 2: LINE X1,Y1-1,X1,0
220 RETURN
```

MTU
Micro Technology Unlimited
2806 Hillsborough Street
P.O. Box 12106
Raleigh, NC 27605 U.S.A.
(919) 833-1458

NOW 80 COLUMN PETS CAN HAVE MTU HIGH RESOLUTION GRAPHICS

MEMSEARCH for the AIM 65

"MEMSEARCH" is a machine language utility program which quickly scans through memory searching for a user-specified sequence. It can assist in locating an ASCII string or an instruction code group. A wild card feature allows for partial matching of sequences up to 16 bytes long.

Bob Kovacs
41 Ralph Road
West Orange, New Jersey 07052

Have you ever had to manually search through memory to look for a certain sequence? Whether you're searching for a particular series of op-codes or ASCII text, doing it with the help of a dump utility or even a disassembler can be painfully slow and prone to error. Clearly this is another job for the computer! The machine language routine described here will accept up to a 16-byte sequence (easily increased if that isn't enough) and identify the starting locations of any matching sequences within the memory range specified by the user.

Although this program was specifically written for use on the AIM 65, using existing monitor routines whenever possible, it shouldn't be too difficult to adapt it to any other 65XX system.

The Program

The flow diagram in figure 1 defines the major events and decision points in memory search routine. Entry point labels are also included to relate these functions to the implementation (see program listing in figure 2).

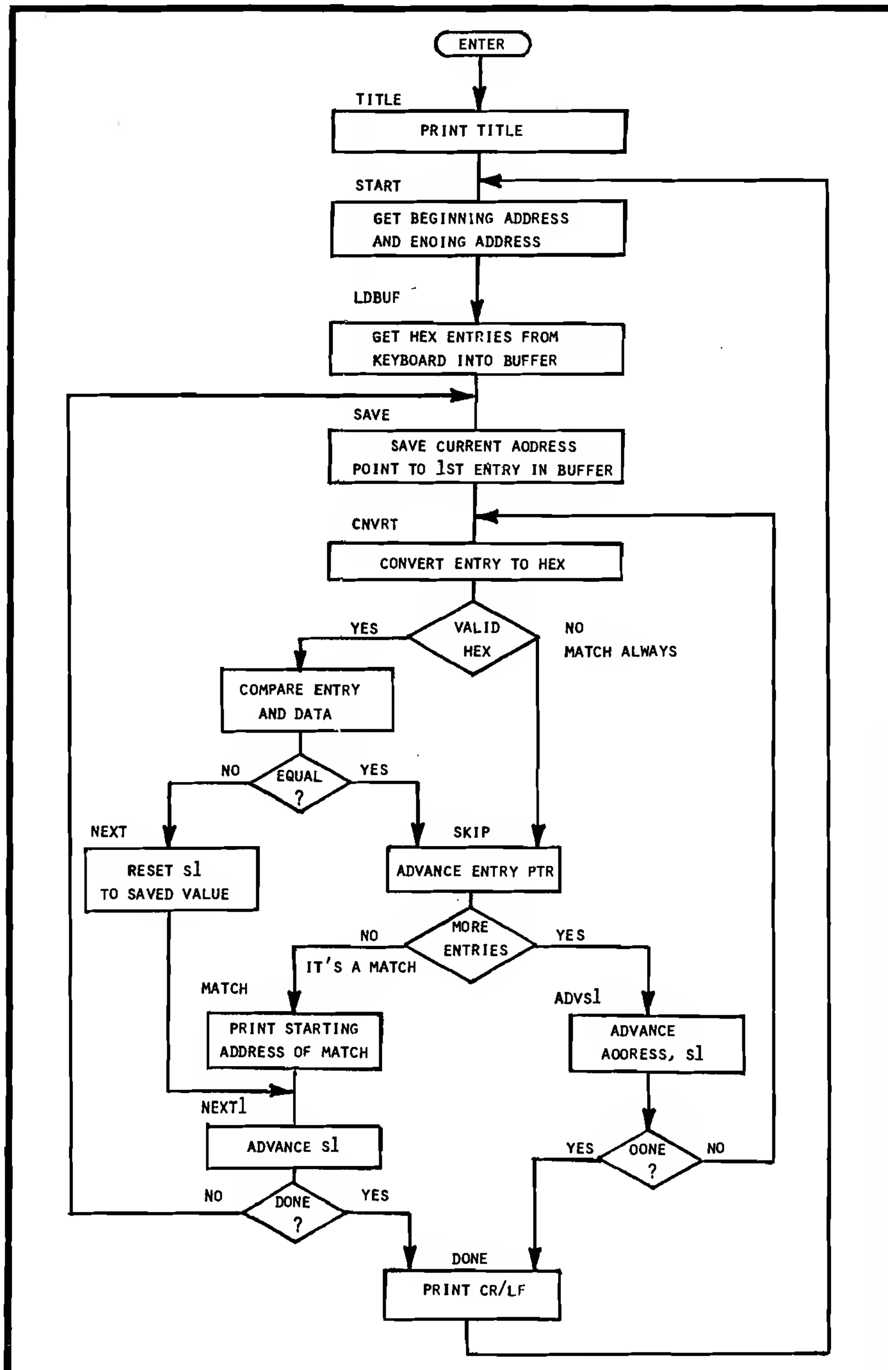


Figure 1: **MEMSEARCH Flow Diagram**

The program begins by establishing a memory search range and the data sequence to be found. This sequence is stored in a buffer using keyboard entry format (2 ASCII bytes per hex byte entry) and is converted to its numerical equivalent each time it is compared to memory. Although this approach is not terribly efficient, it was necessary in order to allow for wild card or don't-care entries, and still permit all 256 possible byte values for valid comparisons. I'm sure that other approaches could have been used to speed up execution time somewhat, but this method is still fairly fast. The worst case of a search through 4K of memory (when all but the 16th entry always match) takes about 6 seconds to complete.

The main body of the program operates by comparing the entry sequence to the data within the memory bounds specified by the user. This is performed one byte at a time, starting with the first entry and then searching for a corresponding value in memory. If a match is found, then the second entry is compared to the contents of the next memory location only. This operation is repeated, always comparing the next entry with the data in the next memory location. If successive successful comparisons exhaust the total number of entries in the buffer, then the entire entry sequence has been matched. At this point the memory address corresponding to the first entry is output, and the search continues at the memory location following the matched sequence.

If at any point an entry fails to match the contents of memory, then the starting address corresponding to the first entry is incremented by one, and the entire operation begins again.

A number of monitor routines were used in MEMSEARCH to minimize its length (192 bytes program and 36 bytes variable space). A summary of the monitor routines used here is shown in figure 3. Along with the name and entry point is a brief description of what the routine does. Those registers affected by that call to the monitor are also listed.

Figure 2

```
0800              ;
0800              ;
0800              ;
0800              ;
0800              ;MEMSEARCH FOR AIM 65
0800              ; BY BOB KOVACS
0800              ; 41 RALPH ROAD
0800              ; WEST ORANGE, NJ 07052
0800              ;
0800              ;ALL KEYBOARD ENTRIES
0800              ; IN HEXADECIMAL
0800              ;NONHEX ENTRIES--"DON'T CARE"
0800              ;
0800              ;ALL MATCHES RETURN BEGINNING
0800              ; ADDRESS OF SEQUENCE
0800              ;
0400                     ORG $400
0400                     OBJ $800
0400              FROM   EQU $E7A3
0400              TO     EQU $E7A7
0400              MOVE   EQU $F910
0400              LDAY   EQU $EB58
0400              PACK   EQU $EA84
0400              NUMA   EQU $EA46
0400              OUTPUT EQU $E97A
0400              REDOUT EQU $E973
0400              BLANK  EQU $E83E
0400              CRLF   EQU $E9F0
0400              QM     EQU $E7D4
0400              STIY   EQU $A427
0400              ADDR   EQU $A41C
0400              S1     EQU $A41A
0400              ;
0400              ;
0400              ;OUTPUT TITLE
0400 A000         TITLE  LDY #$00
0402 B9BD04       TITLE1 LDA MSG,Y
0405 F006                BEQ TITLE2
0407 207AE9              JSR OUTPUT
040A C8                  INY
040B D0F5                BNE TITLE1
040D 20F0E9       TITLE2 JSR CRLF
0410              ;
0410              ;GET BEGINNING & ENDING ADDRESSES
0410 20A3E7       START  JSR FROM
0413 B0FB                BCS START
0415 203EE8              JSR BLANK
0418 2010F9              JSR MOVE
041B 20A7E7       START1 JSR TO
041E B0FB                BCS START1
0420              ;
0420              ;PROMPT USER FOR HEX INPUT
0420 20D4E7       LDBUF  JSR QM
0423 203EE8              JSR BLANK
0426 A200                LDX #$00
0428              ;
0428              ;GET ENTRY PAIRS & STORE IN BUF
0428              ;EXIT ENTRY MODE WITH CR
0428 2073E9       LDBUF1 JSR REDOUT
042B C90D                CMP #$0D
042D F011                BEQ LDBUF2
042F 9DCA04              STA BUFHI,X
0432 2073E9              JSR REDOUT
0435 9DDA04              STA BUFLO,X
0438 203EE8              JSR BLANK
043B E8                  INX
043C E010                CPX #$10
043E 90E8                BCC LDBUF1
```

```
0440 8EC904      LDBUF2 STX ENTCNT
0443 20F0E9             JSR CRLF
0446             ;
0446             ;SAVE CURRENT ADDRESS
0446 AD1AA4      SAVE   LDA S1
0449 8DC704             STA TEMP1
044C AD1BA4             LDA S1+1
044F 8DC804             STA TEMP2
0452             ;
0452             ;READ BUF & CONVERT TO HEX
0452             ;NON-HEX ACTS AS DON'T CARE
0452 A200               LDX #$00
0454 BDCA04      CNVRT  LDA BUFHI,X
0457 2084EA             JSR PACK
045A B014               BCS SKIP
045C BDDA04             LDA BUFLO,X
045F 2084EA             JSR PACK
0462 B00C               BCS SKIP
0464             ;
0464             ;COMPARE TO DATA AT ACTIVE ADDRESS
0464 A000               LDY #$00
0466 A91A               LDA #$1A
0468 2058EB             JSR LDAY
046B CD29A4             CMP STIY+2
046E D01E               BNE NEXT
0470             ;
0470             ;MATCH OR DON'T CARE
0470 E8          SKIP   INX
0471 ECC904             CPX ENTCNT
0474 B007               BCS MATCH
0476 20A804             JSR ADVS1
0479 B027               BCS DONE
047B 90D7               BCC CNVRT
047D             ;
047D             ;GOT A MATCH!
047D             ;OUT SAVED ADDRESS
047D ADC804      MATCH  LDA TEMP2
0480 2046EA             JSR NUMA
0483 ADC704             LDA TEMP1
0486 2046EA             JSR NUMA
0489 203EE8             JSR BLANK
048C D00C               BNE NEXT1
048E             ;
048E             ;NO OR PARTIAL MATCH
048E             ;BACK-UP ACTIVE ADDRESS
048E ADC704      NEXT   LDA TEMP1
0491 8D1AA4             STA S1
0494 ADC804             LDA TEMP2
0497 8D1BA4             STA S1+1
049A 20A804      NEXT1  JSR ADVS1
049D B003               BCS DONE
049F 4C4604             JMP SAVE
04A2             ;
04A2             ;NO MORE DATA--START AGAIN
04A2 20F0E9      DONE   JSR CRLF
04A5 4C1004             JMP START
04A8             ;
04A8             ;COMPARE & BUMP ADDRESS PTR
04A8 AD1AA4      ADVS1  LDA S1
04AB CD1CA4             CMP ADDR
04AE AD1BA4             LDA S1+1
04B1 ED1DA4             SBC ADDR+1
04B4 EE1AA4             INC S1
04B7 D003               BNE ADV
04B9 EE1BA4             INC S1+1
04BC 60          ADV    RTS
04BD             ;
04BD             ;
```

(continued)

Using the Program

Load MEMSEARCH through the assembler using the listing in figure 2. Save the program on tape using the 'DUMP TO TAPE' command from $400 to $4BF.

After loading MEMSEARCH, begin its execution using the '*' and 'G' commands. The beginning address and ending address +1 are entered in response to the 'FROM' and 'TO' prompts. The sequence to be found is entered following the '?' prompt. Values are in hex notation without spaces between bytes (spaces are automatically inserted). Two characters must be entered per byte, and up to 16 bytes can be specified. Non-hex entries act as wild cards and match anything. Terminate the sequence (if less than 16 bytes) with a carriage return. The addresses of any matching data sequences in memory are output and the program loops back to search a new memory block.

Applications

What can MEMSEARCH be used for? Well, everyone has his own needs. I was prompted to write MEMSEARCH in order to locate certain entry points and page zero usage in the AIM 65 BASIC interpreter. Unfortunately Rockwell hasn't provided much information in this area. Nevertheless, I suspected that this was a version of Microsoft BASIC similar to the one known as Applesoft (used in the Apple II). Although quite a bit is known about Applesoft, the memory locations used in the Apple and AIM weren't necessarily the same. Thus the code wouldn't be the same (hence the need for a wild card). With the help of MEMSEARCH I was able to identify the required entry points and page zero locations in a minimum of time.

Bob Kovacs is an electro-optics engineer at Bendix where he is currently responsible for the development of a charge-transfer imaging system used for celestial navigation. He is using an AIM 65 for imager sequencing, data collection and processing in the evaluation of a breadboard system. At home, Bob is involved with hardware/software projects on his Apple II. He also enjoys skiing, gardening and photography.

MICRO Club Circuit

(Continued from page 15)

Apple PI Computer User's Group
Rod Nelson, President, William T. Davis Secretary preside over this club boasting a membership of 276. Meetings are held on the first Thursday of each month at 7:00 pm, at the Colorado School of Mines, Golden, CO. The group meets to help each other learn and enjoy computing with Apples. Contact:

Secretary
P.O. Box 17467
Denver, CO 80217

Las Cruces Computer Club
This dual Apple/TRS-80 users group meets on the first Thursday of each month at 7:30 pm at the SW Computer Center (121 Wyatt Suite 7, Las Cruces, NM 88001). Leonard Fetterhoff is club president for 25 members. For further information contact the club secretary:

John Martellaro
2929 Los Amigos Ct. Apt. B
Las Cruces, NM 88001

Original Apple Corps
Kip J. Reiner is president for this club of 300 members. Meetings are held on the second Sunday of the month at noon at UCLA campus, Young Hall, Room 2224, Los Angeles, CA. This group publishes a club magazine, "Applesauce" for $15.00 a year. They meet to expand the knowledge of Apple computers, hardware and software. For further information, write:

Secretary
19041-2 Hamlin Street
Reseda, CA 91335

Apple-Can
This 200+ membership club meets at 7:30 on the first Wednesday of each month, currently at Forest Hill Public Library. Louis H. Milrad is the club president. This club features many guest speakers and promotes the better understanding of the Apple computer, its applications and limitations. They publish a bimonthly newsletter. Many active subgroups in Telecommunications, Medical, Pascal, Forth, Introduction to BASIC, Games, Business, etc, all with an extensive program library. For further information, contact:

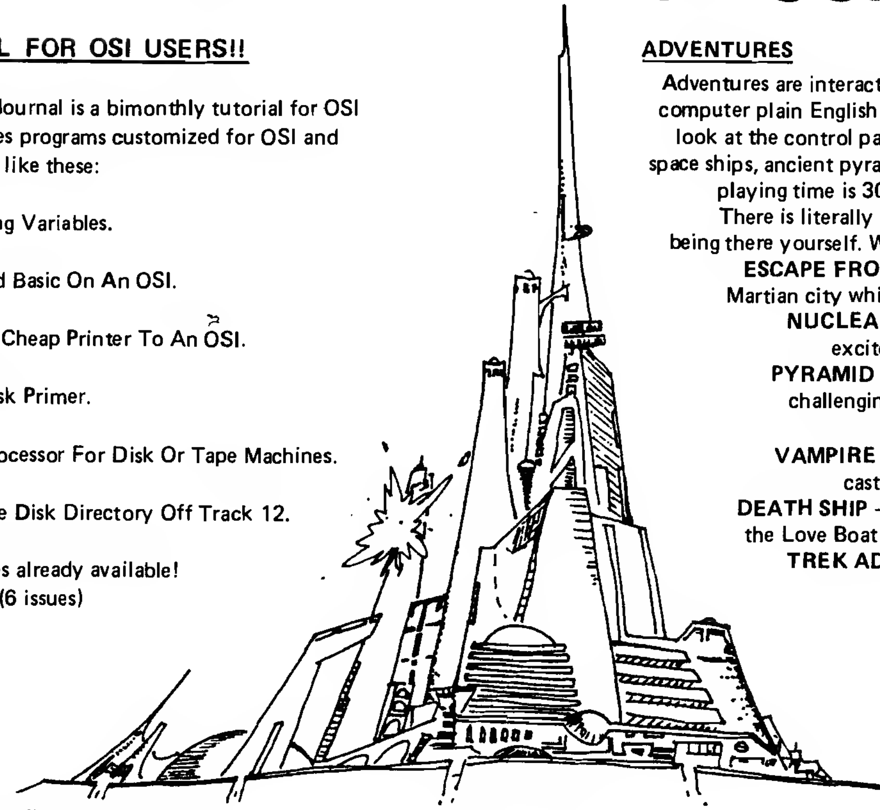
Secretary
Suite 204
2 Gloucester Street
Toronto, Ontario, CANADA
M4Y 1L5

```
04BD 4D454D  137  MSG     ASC 'MEMSEARCH'
04C0 534541
04C3 524348
04C6 00      138          BYT $00
04C7 00      139  TEMP1   BYT $00
04C8 00      140  TEMP2   BYT $00
04C9 00      141  ENTCNT  BYT $00
04CA 313233  142  BUFHI   ASC '12345678'
04CD 343536
04D0 3738
04D2 313233  143          ASC '12345678'
04D5 343536
04D8 3738
04DA 313233  144  BUFLO   ASC '12345678'
04DD 343536
04E0 3738
04E2 313233  145          ASC '12345678'
04E5 343536
04E8 3738
```

Name	Address	Registers Changed	Description
FROM	E7A3	A,X,Y	Output 'FROM' prompt; user inputs 4 character hex address (ESC & DEL are active) which is stored @ADDR. Carry set if non-hex value entered.
TO	E7A7	A,X,Y	Same as FROM except for prompt.
REDOUT	E973	A	Return with a single character from keyboard in accumulator. Echo to output device unless CR input.
MOVE	F910	A,X	2-byte move from ADDR to S1.
LDAY	EB58	A	Performs a LDA (S1),Y without using page zero. Enter with accumulator pointing to S1 via offset from $A400 base address.
PACK	EA84	—	Converts ASCII character in accumulator into hex and packs it with previous value (saved in STIY+2). If not hex (i.e. 0-9,A-F) then original character is returned with carry set.
NUMA	EA46	—	Output contents of accumulator as 2 character hex.
OUTPUT	E97A	—	Output ASCII code in accumulator to active output device(s).
BLANK	E83E	A	Output a single space.
CRLF	E9F0	A	Output a carriage return and line feed.
QM	E7D4	A	Output a question mark.

Figure 3: AIM 65 Monitor Routine Summary

SOFTWARE FOR OSI



A JOURNAL FOR OSI USERS!!

The Aardvark Journal is a bimonthly tutorial for OSI users. It features programs customized for OSI and has run articles like these:

1) Using String Variables.
2) High Speed Basic On An OSI.
3) Hooking a Cheap Printer To An OSI.
4) An OSI Disk Primer.
5) A Word Processor For Disk Or Tape Machines.
6) Moving The Disk Directory Off Track 12.

Four back issues already available!
$9.00 per year (6 issues)

ADVENTURES

Adventures are interactive fantasies where you give the computer plain English commands (i.e. take the sword, look at the control panel.) as you explore alien cities, space ships, ancient pyramids and sunken subs. Average playing time is 30 to 40 hours in several sessions. There is literally nothing else like them — except being there yourself. We have six adventures available.

ESCAPE FROM MARS — Explore an ancient Martian city while you prepare for your escape.
NUCLEAR SUBMARINE — Fast moving excitement at the bottom of the sea.
PYRAMID — Our most advanced and most challenging adventure. Takes place in our own special ancient pyramid.
VAMPIRE CASTLE — A day in old Drac's castle. But it's getting dark outside.
DEATH SHIP — It's a cruise ship — but it ain't the Love Boat and survival is far from certain.
TREK ADVENTURE — Takes place on a familiar starship. Almost as good as being there.

$14.95 each

NEW SUPPORT ROMS FOR BASIC IN ROM MACHINES

C1S — for the C1P only, this ROM adds full screen edit functions (insert, delete, change characters in a basic line.), Software selectable scroll windows, two instant screen clears (scroll window only end full screen.), software choica of OSI or standard keyboerd format, 8ell support, 600 Baud cassette support, and a few other feetures. It plugs in in place of the OSI ROM. NOTE: this ROM also supports video conversions for 24, 32, 48, or 64 characters par lina. All thet end it sells for a measly $39.95.
C1E/C2E for C1/C2/C4/C8 Basic in ROM machines.
This ROM adds full screen editing, software selectable scroll windows, keyboard correction (software selectable), and contains an extended machine code monitor. It has breakpoint utilities, mechine code load and save, block memory move and hex dump utilities. A must for the machine code programmer repleces OSI support ROM. Specify system $59.95

DISK UTILITIES

SUPER COPY — Single Disk Copier
This copy program makes multiple copies, copies track zero, and copies all the tracks that your memory can hold at one time — up to 12 tracks at a pass. It's almost as fast as dual disk copying. — $15.95
MAXIPROSS (WORD PROCESSOR) — 65D polled keyboard only - has global and line edit, right and left margin justification, imbedded margin commands, choice of single, double or triple spacing, file access capabilities and all the features of a major word processor — and it's only $39.95.

P.C. BOARDS

MEMORY BOARDS!! — for the C1P. — and they contain perallel ports!
Aardvarks new memory boerd supports 8K of 2114's and has provision for e PIA to giva a parallel ports! It sells as a bere board for $29.95. When essembled, the board plugs into the expansion connector on the 600 board. Available now!

PROM BURNER FOR THE C1P — Burns single supply 2716's. Bare boerd — $24.95.

MOTHER BOARD — Expand your expansion connector from one to five connectors or use it to adept our C1P boerds to your C4/8P. - $14.95.

ARCADE AND VIDEO GAMES

ALIEN INVADERS with machine code moves — for fast action. This is our best invaders yet. The disk version is so fast that we had to add selectable speeds to make it playable.
Tape - $10.95 — Disk - $12.95

TIME TREK (8K) — real time Startrek action. See your torpedoes move across the screen! Real graphics — no more scrolling displays. $9.95

STARFIGHTER — a real time space war where you face cruisers, battleships and fighters using a variety of weapons. Your screen contains working instrumentation and a real time display of the alien ships. $6.95 in black and white - $7.95 in color and sound.

MINOS — A game with amazing 3D graphics. You see a maze from the top, the screen blanks, end then you are in the maze at ground level, finding your way through on foot. Realistic enough to cause claustrophobia. — $12.95

SCREEN EDITORS

These programs all allow the editing of basic lines. All assume that you ere using the standard OSI video display end polled keyboard.
C1P CURSOR CONTROL — A program that uses no RAM normally available to the system. (We hid it in unused space on page 2). It provides real backspace, insert, delete and replace functions and an optionel instant screen clear.
$11.95
C2/4 CURSOR. This one uses 366 BYTES of RAM to provide a full screen editor. Edit and chenge lines on any part of the screen. (Basic in ROM systems only.)
FOR DISK SYSTEMS — (65D, polled keyboard and standard video only.)
SUPERDISK. Contains a basic text editor with functions similar to the above programs and also contains a renumberer, variable table maker, search and new BEXEC* programs. The BEXEC* provides a directory, create, delete, and change utilities on one track and is worth having by itself. — $24.95 on 5" disk - $26.95 on 8".

AARDVARK IS NOW AN OSI DEALER!

Now you can buy from people who can support your machine.

—THIS MONTH'S SPECIALS—

Superboerd II	$279
C1P Model II	429
C4P	749

. . . and we'll include a free Text Editor Tape with each machine!

Video Modification Plans and P.C. Boards for C1P as low as $4.95

This is only a partial listing of what we hava to offar. We now offer ovar 100 progrems, data sheets, ROMS, and boerds for OSI systems. Our $1.00 cetalog lists it all and contains free program listings and programming hints to boot.

Aardvark Technical Services • 1690 Bolton • Walled Lake, MI 48088
(313) 669-3110

Joysticks for the OSI C4

You can do better than to buy OSI joysticks for the C4P. Here's how to make and test your own.

Charles Platt
P.O. Box 556
New York, New York 10011

The Ohio Scientific Challenger C4 is designed for use with joysticks, which are available from the manufacturer as an optional extra. Anyone who really enjoys playing and/or programming video games will want to take advantage of this feature, since it is much easier to control a game with joysticks than by pressing keys on the keyboard. However, Ohio Scientific joysticks are expensive, not as strong as one would like, and often out of stock at one's local OSI dealer. Having learned these hard facts of life, I decided to take matters into my own hands. Either I would convert Atari joysticks (which are very widely available and not too highly priced), or I would make my own.

Joystick Operation

During a game program the computer needs to know in which direction each joystick is being pushed by the player(s), and whether the "action key" on each joystick is being pressed.

Inside the joystick assembly are four switches which close, one at a time, when the stick is pushed up, down, left, or right. If the stick is pushed diagonally, two of the four switches close simultaneously. In addition there is a fifth switch which is closed when the player pushes the action key.

The computer detects these switch closures via a POKE command in the game program. For example, POKE 57088,128 directs the computer's attention to Joystick A. If the program next asks for a PEEK c emory location 57088, this will yield a number which corresponds to which switches are closed inside the joystick assembly. This routine is similar to a keyboard PEEK routine, and the joysticks can be thought of as extensions to the keyboard.

There is a chart on page 90 of the old C4 user's manual, giving the possible PEEK numbers and the joystick positions which they represent. Unfortunately, the column headings in this chart are incorrect. The figures in the columns headed "Action Key Depressed" are in fact produced when the action key is *not* depressed, while the figures in the "Action Key Not Depressed" column are in fact produced when the action key *is* depressed. If you write your own game program it is important to bear this in mind. This has been corrected in the 1981 version of the C4P User's Manual.

Connecting Non-C4 Joysticks

There are four trapezoidal sockets on the back panel of the C4, adjacent to the fan. The top socket is for Joystick A, the next one down is for Joystick B. (The other two sockets are for keypads A and B.)

Neither the C4 user's manual nor the maintenance manual gives precise information about which pin does what, in the joystick sockets. However, some trial-and-error tests revealed the functions that follow.

Looking at the outside of the socket, numbering the pins from left to right, the top row of pins in each socket can be labelled pins 1 through 5, and the bottom row, pins 6 through 9. In this case, pin 1 is the ground, pin 2 connects to the action key, pin 3 connects to the "Left" switch in the joystick, pin 4 goes to the "Down" switch, pin 5 to the "Right" switch, and pin 6 to the "Up" switch. Pins 7, 8, and 9 are unconnected.

My first experiment was with Atari joysticks. By a rare fluke of standardization in the computer industry, the Atari joystick plug exactly fits the C4 joystick socket. Moreover, the switching inside the Atari joystick unit is similar to the switching of C4 joysticks, and there are precisely six wires in the Atari connecting cable—just right for the six active pins in the C4 socket.

There is one snag however. If you look in the holes in the Atari plug, you will find that not all of them contain metal connectors. Some holes are not used and do not connect to anything. Unfortunately, these holes correspond with pins in the C4 socket which *are* used and *must* be connected to something. So you have to slice open the molded plastic Atari plug to get at the metal connectors, which must be reshuffled into the right sequence, leaving holes 7, 8, and 9 empty to correspond with unused C4 pins 7, 8, and 9.

The Atari wires are color coded and should be matched to the C4's pin numbers as follows:

Black	Pin 1
Orange	Pin 2
Green	Pin 3
Blue	Pin 4
Brown	Pin 5
White	Pin 6

Once you have opened the plug and extracted the little metal connectors which slide onto the pins in the joystick socket (some connectors may be torn loose in the plug-opening operation and will need to be resoldered to their wires), you can slide these connectors individually onto their separate pins, and separate them with small pieces of electrical tape to prevent accidental shorts. You can then test the joystick, using the procedure described later in this article.

When you're sure the joystick is working properly, and all your connections have been made correctly, you can drip some quick-setting epoxy over the metal connectors to encapsulate

them. When the epoxy is dry, the connectors can be slipped off the pins in one unit. The epoxy has, in fact, created a new "plug" around the connectors, to replace the original plug which had to be sliced open.

Making Your Own Joysticks

After using Atari joysticks for a while, I became dissatisfied with their response and decided to build my own. This turned out to be extremely simple.

Each joystick unit consists of a box with a wooden top and bottom and aluminum sides. (The thin aluminum is bent around the wood and nailed to it.) The stick is pivoted where it is screwed to the bottom of the box; some self-centering action is provided by a small compression spring. The stick protrudes through a 1" square hole in the top of the box. Arranged around this hole, screwed to the underside of the top of the box, are four microswitches, positioned so that their contact buttons are just touching the four sides of the stick (which has a square cross-section at this point). Lastly, a pushbutton is mounted on the outside of the top of the box to serve as the action key.

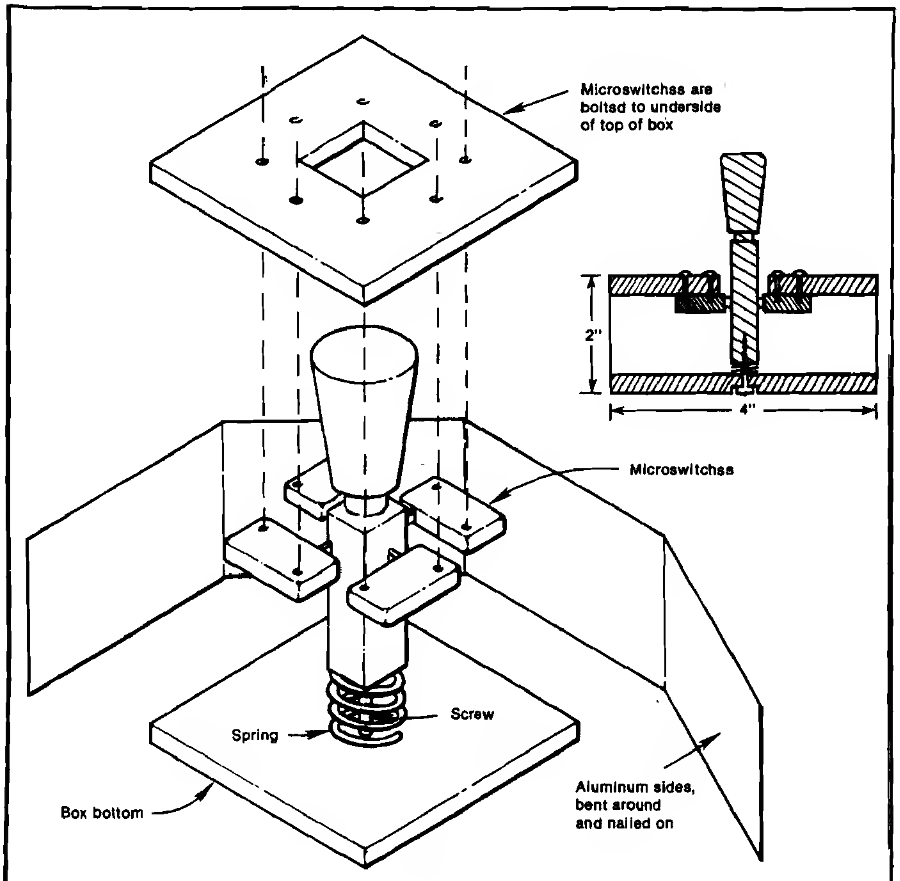


Figure 1

Using microswitches allows a much more positive "feel" than is available from the Atari joysticks. My home-made units provide much more precise control of video games.

Checking Joystick Operation

To make sure you have wired your home-made or Atari joysticks correctly, you can run a simple "POKE and PEEK" test program.

Program for Joystick A

```
10 POKE 2073,96:REM —
   DISABLES CONTROL-C.
   THE ROUTINE WON'T
   FUNCTION TILL YOU DO
   THIS.
20 FOR K=1 TO 200:REM —
   SEE NOTE BELOW
30 POKE 57088,128:REM —
   ACTIVATES JOYSTICK A
40 P=PEEK(57088) AND
   31:REM — PEEK JOYSTICK A
50 PRINT P
60 FOR D=1 TO 200:NEXT
   D:REM — DELAY LOOP
70 NEXT K
80 END
```

Since "Control-C" has been disabled in the program, there is no way of stopping the program once it has started, short of hitting the Break key. So a loop is used, incrementing K by 1 in each of 200 cycles. The program ends at the end of the loop. A delay loop is also used, to stop the figures from racing uncontrollably across the screen.

Note: line 30, the POKE command, is inside the K loop. This is because you must POKE 57088 again after each time you have PEEKed it and it has yielded data. If you write a program which repeatedly PEEKs 57088 for data and does not re-POKE it each time, you will find that the joysticks won't work properly. For a demonstration of this fault, you can run the sample program listed on page 93 of the old C4 user's manual, or pages 45-47 in the new manual. This program erroneously fails to POKE 57088 after PEEKing it. Consequently, as listed, the program doesn't work.

When you test Joystick A, using the test program shown here, you should find that moving the stick generates, on the video screen, the various numbers listed on page 90 of the old manual, page 43 in the new manual. If the numbers are as listed, but they appear in the wrong sequence, you've probably made an incorrect connection in the joystick socket. If the numbers on the screen do not in any way match the numbers in the manual, you have probably made a programming error. Be sure that your PEEK command is PEEK(57088) AND 31. Without the "AND 31" it won't work.

If you are using Atari joysticks and you find that pushing the stick directly up and down, or from side to side, produces numbers which wrongly indicate diagonal motion, the problem is simply that you are pushing the stick too hard, thus turning on two switches instead of only one at a time. Only very light pressure is required.

Once you have tested Joystick A, you can test Joystick B by rewriting two lines of the test program:

```
30 POKE 57088,16
40 P=PEEK(57088) AND 248
```

These are the POKE and PEEK which give access to Joystick B.

Happy game playing!

MICRO

Apple Memory Maps, Part 1

Your Apple can draw accurate memory maps of Integer BASIC and Applesoft programs, together with their associated variables, arrays, and strings, by using the information contained in various pointers. DOS, MAXFILES, and RAM Applesoft can also be displayed.

Peter A. Cook
1443 N. 24th Street
Mesa, Arizona 85203

This article will be presented in two parts. Part 1 contains examples of memory maps produced by the Apple, which show where the computer stores programs in its memory. Part 2 will contain the "Memory Map" program listing and a description of how it works.

Memory maps show where computers store data in their memories. A 48K Apple actually has 65,536, or 64K, memory locations in which data can be stored. Locations 0 to 49151, the first 48K, are available for storing changeable information (Random Access Memory), while locations 49152 to 65535, the last 16K, are for permanently-installed data which can't be changed by the user (Read Only Memory). The computer places data into specific locations in the RAM memory area, depending on what type data it is, and which language is being used.

Various charts in the Apple reference manuals show where programs are stored in RAM, along with their associated variables, arrays and strings. The disk and cassette versions of Applesoft are also stored in this area, and so is the Disk Operating System and its file buffers. The charts are adequate for simple programs, but for more complex ones you need to know exactly how much space is used by the different program components. This is especially important if the Hi-Res graphics pages are used, or if machine language subroutines are included.

<table>
<tr><th rowspan="2">APPLE II</th><th colspan="2">Applesoft Card</th><th rowspan="2">NO CARD</th></tr>
<tr><th>Switch up</th><th>Switch down</th></tr>
<tr><td>DOS 'INT'</td><td>Integer BASIC</td><td>Integer BASIC</td><td>Integer BASIC</td></tr>
<tr><td>DOS 'FP'</td><td>ROM Applesoft</td><td>ROM Applesoft</td><td>RAM Applesoft (disk)</td></tr>
<tr><td rowspan="2">no DOS</td><td rowspan="2">ROM Applesoft</td><td>Integer BASIC</td><td>Integer BASIC</td></tr>
<tr><td>RAM Applesoft (cassette)</td><td>RAM Applesoft (cassette)</td></tr>
</table>

Figure 1: Language availability for various configurations of the Apple II.

Description

The following maps cover the Apple's RAM memory area from 2048 to the highest available RAM location in your machine. The area from 0 to 2047 is not included because it is used by the computer for various internal functions and is not generally available for BASIC programs.

The MEMORY MAP program will provide the following information:

1. Maximum amount of RAM available.
2. Whether or not DOS has been booted.
3. Number of DOS file buffers reserved (MAXFILES).
4. Current language in use.
5. Whether or not a program has been loaded, or run.
6. Location and length of program, variables, arrays, and strings.
7. Amount of free space remaining.
8. Setting of LOMEM and HIMEM.
9. Location of Hi-Res graphics pages.
10. Extent of the "garbage collection" of old strings.

The program will work with all versions of the Apple II or Apple II Plus, ranging in size from 16K to 48K, with either the old monitor ROM or the new autostart ROM installed. It will accept programs from cassette as well as disk, but it will only work with DOS version 3.2.

The program was designed for use with Integer BASIC, RAM Applesoft, or ROM Applesoft. However, I do not know what the results will be if you use these languages with the Pascal language system installed. Language availability for various configurations of the Apple II is shown in figure 1.

Integer BASIC Memory Maps

The following examples show how the MEMORY MAP program can be used with Integer BASIC programs. We will use a 48K Apple II with Applesoft card in slot 0, printer in slot 2, and disk drive in slot 6. For the first example, turn on the computer without loading DOS. Enter the monitor and load the

```
----------------------------------------------------------
!                                                        !
! MEMORY MAP:  INTEGER BASIC                             !
!                                                        !
!            !------------! HM 49152   $C000             !
!            !            !                              !
!            !            !                              !
!            !            !                              !
!            !            !                              !
! HIRES      !            !                              !
!            !            !                              !
! 24576      !            !                              !
! $6000      ! FREE SPACE !    47104   $B800             !
!            !            !                              !
! 16384      !            !                              !
! $4000      !            !                              !
!            !            !                              !
!  8192      !            !                              !
! $2000      !            !                              !
!            !            !                              !
!            !            !                              !
!            !            !                              !
!            !            !                              !
!            !------------! LM   2048  $0800             !
!                                                        !
----------------------------------------------------------
```

Figure 2: **Integer BASIC map with nothing in memory.**

```
----------------------------------------------------------
!                                                        !
! MEMORY MAP:  INTEGER BASIC                             !
!                                                        !
!            !------------! HM 49152   $C000             !
!            !            !                              !
!            !            !                              !
!            !            !                              !
!            !            !                              !
! HIRES      !            !                              !
!            !            !                              !
! 24576      !            !                              !
! $6000      ! FREE SPACE !    47098   $B7FA             !
!            !            !                              !
! 16384      !            !                              !
! $4000      !            !                              !
!            !            !                              !
!  8192      !------------!     2054   $0806             !
! $2000      ! VARIABLES  !                              !
!            !            !                              !
!            !   ARRAYS   !                              !
!            !            !                              !
!            !  STRINGS   !                              !
!            !------------! LM   2048  $0800             !
!                                                        !
----------------------------------------------------------
```

Figure 3: **Integer BASIC map showing area for variables.**

```
MEMORY MAP:   INTEGER BASIC

                  !------------! HM 49152   $C000
                  !  PROGRAM   !
                  !------------!     49138   $BFF2
                  !            !
                  !            !
 HIRES            !            !
                  !            !
 24576            !            !
 $6000            ! FREE SPACE !     47090   $B7F2
                  !            !
 16384            !            !
 $4000            !            !
                  !            !
  8192            !            !
 $2000            !            !
                  !            !
                  !            !
                  !            !
                  !            !
                  !------------! LM   2048   $0800
```

***Figure 4:* Integer BASIC map showing program area.**

```
MEMORY MAP:   INTEGER BASIC

                  !------------! HM 49152   $C000
                  !  PROGRAM   !
                  !------------!     49138   $BFF2
                  !            !
                  !            !
 HIRES            !            !
                  !            !
 24576            !            !
 $6000            ! FREE SPACE !     47084   $B7EC
                  !            !
 16384            !            !
 $4000            !            !
                  !            !
  8192            !------------!      2054   $0806
 $2000            ! VARIABLES  !
                  !            !
                  !   ARRAYS   !
                  !            !
                  !  STRINGS   !
                  !------------! LM   2048   $0800
```

***Figure 5:* Integer BASIC map after running program.**

MEMORY MAP machine language program from cassette. Enter BASIC and use CALL 13000 to run it. More detailed loading instructions will be provided in Part 2, along with the actual program itself.

```
3200.38DFR
Control B
CALL 13000
```

Figure 2 shows that LOMEM is set at 2048, HIMEM is set at 49152, and 47,104 bytes of free space are waiting to be used.

Now define a simple variable and call MEMORY MAP again.

```
A = 1
CALL 13000
```

Figure 3 shows that the variable is stored just above LOMEM and contains 6 bytes, as do all simple variables in Integer BASIC.

Clear the variable and enter the same statement in the form of a program.

```
Reset, Control B
10 A = 1
20 END
CALL 13000
```

Notice that the program has been stored just below HIMEM, as shown in figure 4.

Load the program again, and this time run it to see what happens.

```
Reset, Control B
10 A = 1
20 END
RUN
CALL 13000
```

The program creates the same variable in figure 5 that was entered in figure 3.

Now load the DOS. Type INT to remove the greeting program, then re-enter the above program and run it. The DOS boot will clobber MEMORY MAP, so it too will have to be reloaded. Now that we have DOS, we can use BRUN MEMORY MAP instead of the separate commands for loading and calling 13000.

```
Reset, Control B
PR#6
INT
10 A = 1
20 END
RUN
BRUN MEMORY MAP
```

Figure 6 shows the large amount of space used by the DOS and its file buffers. The default number of buffers, three, has been reserved and HIMEM has been reset to 38400.

For a more complex case, let's reserve the maximum number of file buffers, 16, drastically change the values of LOMEM and HIMEM, and run our same program again. Be aware that LOMEM: and HIMEM: are not legal Integer BASIC commands, but can be used with DOS.

```
INT
MAXFILES 16
LOMEM: 14000
HIMEM: 15000
10 A = 1
20 END
RUN
BRUN MEMORY MAP
```

The memory map in figure 7 shows that everything has been set as specified. Note the small amount of free space remaining.

```
MEMORY MAP:  INTEGER BASIC

            !------------!       49152   $C000
            !    DOS,    !
            ! FILES (3)  !
            !------------!  HM 38400   $9600
            !  PROGRAM   !
HIRES       !------------!       38386   $95F2
            !            !
24576       !            !
$6000       ! FREE SPACE !       36332   $8DEC
            !            !
16384       !            !
$4000       !            !
            !            !
 8192       !------------!        2054   $0806
$2000       ! VARIABLES  !
            !            !
            !   ARRAYS   !
            !            !
            !  STRINGS   !
            !------------!  LM  2048   $0800
```

Figure 6: Integer BASIC map showing DOS and program.

```
MEMORY MAP:  INTEGER BASIC

            !------------!       49152   $C000
            !    DOS,    !
            ! FILES (16) !
            !------------!       30665   $77C9
            !------------!  HM 15000   $3A98
HIRES       !  PROGRAM   !
            !------------!       14986   $3A8A
24576       !            !
$6000       ! FREE SPACE !         980   $03D4
            !            !
16384       !            !
$4000       !            !
            !------------!       14006   $36B6
 8192       ! VARIABLES  !
$2000       !            !
            !   ARRAYS   !
            !            !
            !  STRINGS   !
            !------------!  LM 14000   $36B0
            !------------!        2048   $0800
```

Figure 7: Integer BASIC map with changed LOMEM and HIMEM.

Applesoft Memory Maps

Applesoft stores everything quite differently than does Integer BASIC. To demonstrate, type FP to change languages and clear the preceding program, then call MEMORY MAP.

```
FP
CALL 13000
```

Figure 8 shows that the program storage area is now at the bottom of memory instead of at the top. With no program loaded, the program pointer starts at 2049 and the end of program pointer starts one or two bytes higher. LOMEM is set above the program. Location 2048 contains a "0" because each program line must be preceded by a zero.

In Applesoft, the variables, arrays, and strings are all stored in separate areas instead of in the one combined area used by Integer BASIC. We can see this by creating some simple examples and looking at the result with MEMORY MAP.

```
A = 1
DIM B(10)
C$ = "STRING"
CALL 13000
```

```
------------------------------------------------------------
!                                                          !
!    MEMORY MAP:   APPLESOFT                               !
!                                                          !
!                  !-------------!     49152   $C000       !
!                  !    DOS,     !                         !
!                  ! FILES (3)   !                         !
!                  !-------------!  HM 38400   $9600       !
!                  !             !                         !
!    HIRES         !             !                         !
!                  !             !                         !
!    24576         !             !                         !
!    $6000         ! FREE SPACE  !     36349   $8DFD       !
!                  !             !                         !
!    16384         !             !                         !
!    $4000         !             !                         !
!                  !             !                         !
!     8192         !             !                         !
!    $2000         !             !                         !
!                  !             !                         !
!                  !-------------!  LM  2051   $0803       !
!                  !   PROGRAM   !                         !
!                  !-------------!      2049   $0801       !
!                  !-------------!      2048   $0800       !
!                                                          !
------------------------------------------------------------
```

***Figure 8:* Applesoft map with only the DOS in memory.**

```
------------------------------------------------------------
!                                                          !
!    MEMORY MAP:   APPLESOFT                               !
!                                                          !
!                  !-------------!     49152   $C000       !
!                  !    DOS,     !                         !
!                  ! FILES (3)   !                         !
!                  !-------------!  HM 38400   $9600       !
!                  !   STRINGS   !                         !
!    HIRES         !-------------!     38394   $95FA       !
!                  !             !                         !
!    24576         !             !                         !
!    $6000         ! FREE SPACE  !     36267   $8DAB       !
!                  !             !                         !
!    16384         !             !                         !
!    $4000         !             !                         !
!                  !-------------!      2127   $084F       !
!     8192         !   ARRAYS    !                         !
!    $2000         !-------------!      2065   $0811       !
!                  !  VARIABLES  !                         !
!                  !-------------!  LM  2051   $0803       !
!                  !   PROGRAM   !                         !
!                  !-------------!      2049   $0801       !
!                  !-------------!      2048   $0800       !
!                                                          !
------------------------------------------------------------
```

***Figure 9:* Applesoft map showing variable, array, and string areas.**

Notice in figure 9 that the variables start at LOMEM. Applesoft variables are seven bytes long. The variable area contains 14 bytes, for A and C$. Arrays in Applesoft can be multidimensional, so they are placed in a separate location above the variables. The array space is determined by rules given in the Applesoft reference manual, pages 119 and 137. The string variable C$ is stored in the variable area with a pointer to the word "STRING" in the string area. Note that the string area contains exactly six characters.

Something interesting happens when you put the above statements into the form of an executable program. Clear the memory, type in the program, and look at its memory map to see that the program has indeed been stored. See figure 10.

```
FP
10 A = 1
20 DIM B(10)
30 C$ = "STRING"
CALL 13000
```

Now run the program and look at it again.

```
RUN
CALL 13000
```

Figure 11 shows that the variable area still contains 14 bytes, and that array B is still the same, but there is no string in the string area. This is because the letters of the string are contained in the program area, and the pointer in C$ obtains the string from the program.

Whenever new characters are assigned to the same string variable, they are added to the string area even if they are the same as those already assigned to that variable. A clutter of old strings thus begins to form, known as the "garbage collection." Its formation can be demonstrated by entering the same statement several times.

```
FP
A$ = "STRING"
A$ = "STRING"
A$ = "STRING"
CALL 13000
```

Notice in figure 12 that there are now 18 bytes stored in the string area, even though only six of them are being used.

The variable area contains seven bytes for A$, the one variable in use.

The Applesoft reference manual makes the following statement on page 53:

> "Applesoft stores duplicate strings only once. That is, if A$ = "PIPPIN" and B$ = "PIPPIN" then the string "PIPPIN" will be stored only once."

Let's try it and see.

```
FP
A$ = "PIPPIN"
B$ = "PIPPIN"
CALL 13000
```

Figure 13 shows that there are 12 bytes in the string area instead of only six. If you enter the monitor mode and examine the variable area you will see that the two string variables point to different locations in the string area. This obviously indicates that Applesoft does *not* store duplicate strings only once.

The actual length of a program doesn't always correspond with the amount of memory required. Just because your program is short doesn't mean you have lots of memory left over. In Applesoft it is easy to create a multidimensional array which uses up all memory space in a 48K machine.

```
FP
DIM A(97,73)
BRUN MEMORY MAP
```

Figure 14 shows that there are only 80 free bytes remaining after dimensioning the array. To verify that MEMORY MAP is indeed providing accurate information, you can check the free space remaining by using the FRE(0) command.

```
PRINT FRE(0)
80
```

If you don't need to use floating point numbers, a good way to save array space is to define the array as an integer array.

```
FP
DIM A%(97,73)
BRUN MEMORY MAP
```

```
-----------------------------------------------------------------
!                                                               !
!   MEMORY MAP:  APPLESOFT                                      !
!                                                               !
!                  !------------!      49152  $C000             !
!                  !    DOS,    !                               !
!                  ! FILES (3)  !                               !
!                  !------------! HM  38400  $9600              !
!                  !            !                               !
!   HIRES          !            !                               !
!                  !            !                               !
!   24576          !            !                               !
!   $6000          ! FREE SPACE !      36314  $8DDA             !
!                  !            !                               !
!   16384          !            !                               !
!   $4000          !            !                               !
!                  !            !                               !
!    8192          !            !                               !
!   $2000          !            !                               !
!                  !            !                               !
!                  !------------! LM   2086  $0826              !
!                  !  PROGRAM   !                               !
!                  !------------!      2049  $0801              !
!                  !------------!      2048  $0800              !
!                                                               !
-----------------------------------------------------------------
```

Figure 10: Applesoft map showing program area.

```
-----------------------------------------------------------------
!                                                               !
!   MEMORY MAP:  APPLESOFT                                      !
!                                                               !
!                  !------------!      49152  $C000             !
!                  !    DOS,    !                               !
!                  ! FILES (3)  !                               !
!                  !------------! HM  38400  $9600              !
!                  !            !                               !
!   HIRES          !            !                               !
!                  !            !                               !
!   24576          !            !                               !
!   $6000          ! FREE SPACE !      36238  $8D8E             !
!                  !            !                               !
!   16384          !            !                               !
!   $4000          !            !                               !
!                  !------------!      2162  $0872              !
!    8192          !   ARRAYS   !                               !
!   $2000          !------------!      2100  $0834              !
!                  ! VARIABLES  !                               !
!                  !------------! LM   2086  $0826              !
!                  !  PROGRAM   !                               !
!                  !------------!      2049  $0801              !
!                  !------------!      2048  $0800              !
!                                                               !
-----------------------------------------------------------------
```

Figure 11: Applesoft map after running program.

```
---------------------------------------------------
!                                                  !
!  MEMORY MAP:  APPLESOFT                          !
!                                                  !
!              !------------!   49152  $C000       !
!              !   DOS,     !                      !
!              ! FILES (3)  !                      !
!              !------------! HM 38400 $9600       !
!              !  STRINGS   !                      !
!  HIRES       !------------!   38382  $95EE       !
!              !            !                      !
!  24576       !            !                      !
!  $6000       ! FREE SPACE !   36324  $8DE4       !
!              !            !                      !
!  16384       !            !                      !
!  $4000       !            !                      !
!              !            !                      !
!   8192       !            !                      !
!  $2000       !------------!    2058  $080A       !
!              ! VARIABLES  !                      !
!              !------------! LM 2051  $0803       !
!              !  PROGRAM   !                      !
!              !------------!    2049  $0801       !
!              !------------!    2048  $0800       !
!                                                  !
---------------------------------------------------
```

Figure 12: **Applesoft map showing formation of "garbage collection" for one string variable.**

```
---------------------------------------------------
!                                                  !
!  MEMORY MAP:  APPLESOFT                          !
!                                                  !
!              !------------!   49152  $C000       !
!              !   DOS,     !                      !
!              ! FILES (3)  !                      !
!              !------------! HM 38400 $9600       !
!              !  STRINGS   !                      !
!  HIRES       !------------!   38388  $95F4       !
!              !            !                      !
!  24576       !            !                      !
!  $6000       ! FREE SPACE !   36323  $8DE3       !
!              !            !                      !
!  16384       !            !                      !
!  $4000       !            !                      !
!              !            !                      !
!   8192       !            !                      !
!  $2000       !------------!    2065  $0811       !
!              ! VARIABLES  !                      !
!              !------------! LM 2051  $0803       !
!              !  PROGRAM   !                      !
!              !------------!    2049  $0801       !
!              !------------!    2048  $0800       !
!                                                  !
---------------------------------------------------
```

Figure 13: **Applesoft map of two string variables with duplicate strings.**

(continued)

Note the large difference in space required for the array in figure 15 as compared to the preceding one. The addition of a % sign saved 21,756 bytes!

If you don't have ROM Applesoft installed, you must load Applesoft into RAM from either cassette or disk. To demonstrate, turn off the computer and remove the Applesoft card. Turn the computer back on and load Applesoft from disk by typing FP. Then run MEMORY MAP to see where RAM Applesoft is stored.

```
Control B
PR#6
FP
BRUN MEMORY MAP
```

Figure 16 shows that RAM Applesoft is stored below the program area, and that it uses a large amount of space. By referring to the Hi-Res locations on the left, you can see that Hi-Res graphics' page one is not available when using RAM Applesoft.

For our final example, let's create the most complicated map possible by using RAM Applesoft, a different LOMEM and HIMEM, and all types of variables. I also tried to change MAXFILES, but it doesn't seem to work with RAM Applesoft. See figure 17.

```
LOMEM: 15000
HIMEM: 20000
A = 1
DIM B(10)
C$ = "STRING"
CALL 13000
```

That concludes the examples. Next month's article will contain the Memory Map program listing and will describe how it works.

Lieutenant Colonel Pete Cook is a jet pilot instructor at Williams Air Force Base. He is assigned to the Air Force's Human Resources Laboratory, Operations Training Division, a large research facility for designing advanced aircraft simulations, and one of the largest computer complexes in Arizona.

This is his third article for MICRO.

```
MEMORY MAP:  APPLESOFT

                 !------------!      49152   $C000
                 !    DOS,    !
                 ! FILES (3)  !
                 !------------! HM 38400   $9600
                 !            !
HIRES            !            !
                 !            !
24576            !            !
$6000            ! FREE SPACE !         80   $0050
                 !            !
16384            !            !
$4000            !            !
                 !            !
 8192            !            !
$2000            !------------!      38320   $95B0
                 !   ARRAYS   !
                 !------------! LM   2051   $0803
                 !  PROGRAM   !
                 !------------!       2049   $0801
                 !------------!       2048   $0800
```

***Figure 14:* Applesoft map of large floating point array.**

```
MEMORY MAP:  APPLESOFT

                 !------------!      49152   $C000
                 !    DOS,    !
                 ! FILES (3)  !
                 !------------! HM 38400   $9600
                 !            !
HIRES            !            !
                 !            !
24576            !            !
$6000            ! FREE SPACE !      21836   $554C
                 !            !
16384            !            !
$4000            !            !
                 !            !
 8192            !            !
$2000            !------------!      16564   $40B4
                 !   ARRAYS   !
                 !------------! LM   2051   $0803
                 !  PROGRAM   !
                 !------------!       2049   $0801
                 !------------!       2048   $0800
```

***Figure 15:* Applesoft map of large integer array.**

```
MEMORY MAP:  APPLESOFT

             !-----------!     49152  $C000
             !   DOS,    !
             ! FILES (3) !
             !-----------!  HM 38400  $9600
             !           !
 HIRES       !           !
             !           !
 24576       !           !
 $6000       ! FREE SPACE!     26109  $65FD
             !           !
 16384       !           !
 $4000       !           !
             !           !
  8192       !           !
 $2000       !           !
             !-----------!  LM 12291  $3003
             !  PROGRAM  !
             !-----------!     12289  $3001
             ! APPLESOFT !
             !-----------!      2048  $0800
```

Figure 16: **Applesoft map with RAM Applesoft loaded.**

```
MEMORY MAP:  APPLESOFT

             !-----------!     49152  $C000
             !   DOS,    !
             ! FILES (3) !
             !-----------!     38400  $9600
             !-----------!  HM 20000  $4E20
 HIRES       !  STRINGS  !
             !-----------!     19994  $4E1A
 24576       !           !
 $6000       ! FREE SPACE!      4918  $1336
             !           !
 16384       !-----------!     15076  $3AE4
 $4000       !  ARRAYS   !
             !-----------!     15014  $3AA6
  8192       ! VARIABLES !
 $2000       !-----------!  LM 15000  $3A98
             !-----------!     12291  $3003
             !  PROGRAM  !
             !-----------!     12289  $3001
             ! APPLESOFT !
             !-----------!      2048  $0800
```

Figure 17: **Applesoft map showing most complex case.**

MICRO™

AIM SUPPORTWARE BRINGS YOU A FULL SPECTRUM OF SUPPORT for your AIM-65 SYSTEM!

ENCLOSURE, extra board opt. (enclosure gp) . . . . $ 46.
Dram Plus (Computerist) 16K . . . $325 . . . 32K . . . 395.
Video Plus II (Computerist) . . . . 325.
Additional memory, 2114's 300nS. each . . . . 5.
2K dynamic RAM, 4116's 200nS. each . . . . 5.
Power Supply: 5V-10A, 24V-.5A, −12V-2A . . . . 65.
Mother Plus (Computerist) . . . . 125.
Stand alone processor, 2MHz Industrial Qual. . . . . 12.
1MHz Commercial Qual. 8.50
*2MHz processor will not run machine at 2MHz.

PRINTERS: Epson MX-80 9×9 matrix, bidirectional, 80 CPS, enhancement mode, normal or double enhancement mode, adjustable # characters per inch, right hand justification, graphics . . . . $645.

PRINTERS: MI² 7×16, red & black, bidirectional, 180 CPS, tractor feed up to 19 inches . . . . $900.
BASIC . . . . $96 . . . . PL/65 . . . . $125 . . . . FORTH . . . . 146.

AMDEK (Leedex) B/W Monitor . . . . . . Video 100 . . . . . . 169.
Video 100 G (Green Phosphor Tube) . . . . 189.
Video 100-80 (has room for std. disk & 11" 11"×14" PC board for control electronics) . . . . 199.

Coming soon from Amdek — Terminal keyboard 58 key, 2 microprocessor control of video. Has RS-232 and built-in telephone coupler.

FORTH is a high speed interpreter. It runs approximately 10 times faster than BASIC. It is extensible, meaning you can add to it and even change word meaning. (There are no reserved words.) New command words can be as much as 31 char. long, containing any char. but space, rtn. or exc., even in first position. FORTH is modular for fast development and debugging. AIM 65 FORTH has floating point arithmetic, unavailable on some machines. FORTH is the most advanced language available on any 8-bit micro, and comes in two 4K ROM's that plug into the BASIC SLOTS. No disk system is necessary. With FORTH on board, you will definitely be proud to be an AIM owner, and will hopefully add to the growing wealth of software written in FORTH.

AIM Supportware shops around to bring you the best prices. Also available: A run-time monitor for the AIM. 2K in EPROM, $95, or cassette, $70. The Run-time Monitor is a tremendous value when the AIM is used as a controller. It incorporates a task scheduler which allows vectoring to and from sub-routines written in machine code or a higher level language.

And if you don't yet have an AIM, from ROCKWELL INTERNATIONAL, we have the Advanced Interactive Microcomputer with 8K Monitor, 20 char. display and 20 char. printer:
1K RAM . . . . $419.
4K RAM . . . . 499.
4K Dual Pass Assembler . . . . 82.

Coming soon from AIM Supportware:
A versatile extended editor, with convenience foremost. Features Grab & Stash # of lines, paragraph, or page, for move or dup. Homing point. Global search and replace, Improved Change and much more.

Also available from AIM Supportware: Custom software and hardware development. And within Indiana, Applications Engineering consultation.

For detailed information on any of the above, write ***AIM Supportware, P.O. Box 681, 1112 So. FORTH St., Terre Haute, IN 47802 or call (812) 232-6824.***

K I M A I M S Y M T I M

END FRUSTRATION!!

FROM CASSETTE FAILURES PERRY PERIPHERALS HAS THE HDE SOLUTION

OMNIDISK SYSTEMS (5" and 8")

ACCLAIMED HDE SOFTWARE

- **Assembler, Dynamic Debugging Tool, Text Output Processor, Comprehensive Memory Test**
- **Coming Soon—HDE BASIC**

PERRY PERIPHERALS S-100 PACKAGE

Adds Omnidisk (5") to Your KIM/S-100 System

- **Construction Manual—No Parts**
- **FODS & TED Diskette**
- **$20. +$2. postage & handling.** (NY residents add 7% tax) (specify for 1 or 2 drive system)

Place your order with:
PERRY PERIPHERALS
P.O. Box 924
Miller Place, N.Y. 11764
(516) 744-6462

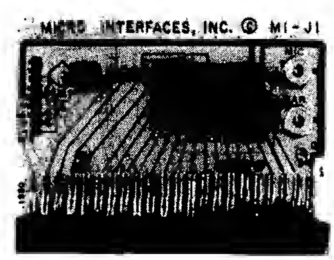
Your Full-Line HDE Distributor/Exporter

For your ***AIM 65***

MI-JI User applications connector

Convenient features

MIC and EAR jacks for cassette recorders
RS232C or 20mAmp serial ports
16 I/O lines + 4 control lines -- VIA signals
Solder pads for remote lines



MICRO INTERFACES, Inc.
P.O. Box 14520
Minneapolis, MN 55414
(612-426-4603)

Assembled, tested 29.95
(90-day warranty)
Bare PC board 13.95

Ask about our Microcomputer DC Control System (MCS) with ROM-based software for real-time control with BASIC

SYM Time-Remaining Timer

This program measures elapsed time and sounds an alarm when the preset limit is reached.

Ralph Orton
16015 San Fernando Mission Blvd.
Granada Hills, California 91344

Here's a neat little timer for a "barebones" 1K SYM that can satisfy a host of timing chores for you. Time phone calls, eggs, exercise sessions and contests. Don't miss TV shows, dates or appointments. My own most frequent use is to keep from "timing out" on the local amateur radio repeaters and thus I avoid the embarrassment associated with such a happening. Operating features include:

1. No peripherals required
2. Two modes of operation
3. *Long* timing internals (more than 4 days) with one second resolution
4. Positive status indications
5. Settable pre-time out warning
6. Ease of operation

Before entering into an explanation of timer operation, I would like to thank John Gieryic for his helpful article in the April, 1979 issue of *MICRO*; "SYM-1 6522-Based Timer." It was John's article that provided the heart of this timer.

Operation is straightforward. After entering the program, enter desired pre-time out warning (hours, minutes, seconds) and desired operating mode. These are entered sequentially in the order indicated above, starting at address "0000". For example, you have decided, as an exercise in self improvement, to restrict yourself to 10½ hours of TV per week. However, you're not crazy about the timekeeping involved

```
0800          ;*************************
0800          ;*                       *
0800          ;* TIME RAMAINING TIMER  *
0800          ;*                       *
0800          ;*   RALPH R. ORTON      *
0800          ;*                       *
0800          ;*************************
0800          ;
0800          ;FOR THE SYM-1
0800          ;SHOWS TIME REMAINING ON READOUT
0800          ;AND SOUNDS A SETTABLE WARNING.
0800          ;DISPLAYS READY WHENEVER TIMER
0800          ;IS STOPPED.
0800          ;
0800          ;TWO MODES OF OPERATION:
0800          ;"RESET" & "ACCUMULTE"
0800          ;
0800          ;LOAD TIMER INTERVAL AND WARNING
0800          ;TIME STARTING AT ADDRESS $0000 AS
0800          ;SHOWN.  LOAD MODE AT $0006. PRO-
0800          ;GRAM STARTS AT $0200.....
0800          ;
0800          HRSSET EPZ $00            ;SET HOURS HERE
0800          MINSET EPZ $01            ;SET MINUTES HERE
0800          SECSET EPZ $02            ;SET SECONDS HERE
0800          HRSWAR EPZ $03            ;WARNING HOURS HERE
0800          MINWAR EPZ $04            ;WARNING MINUTES HERE
0800          SECWAR EPZ $05            ;WARNING SECONDS HERE
0800          MODE   EPZ $06            ;"AC" = ACCUMULATE
0800          HRSREM EPZ $07            ;CURRENT HOURS HERE
0800          MINREM EPZ $08            ;CURRENT MINUTES HERE
0800          SECREM EPZ $09            ;CURRENT SECONDS HERE
0800          FRACNT EPZ $0A            ;20THS OF A SECOND ARE HERE
0800          DISPIN EPZ $0B            ;INPUT FOR "LODISP" SUBROUTINE
0800          DISPOP EPZ $0C            ;STORAGE FOR "LODISP" SUBROUTINE

0800          ;
0800          ; MONITOR SUBROUTINES:
0800          ;
0800          NIBASC EQU $8309
0800          INSTAT EQU $8386
0800          DELAY  EQU $835A
0800          SCAND  EQU $8906
0800          BEEP   EQU $8972
0800          OUTDSP EQU $89C1
0800          ACCESS EQU $8B86
0800          ;
0800          DISBUF EQU $A640
0800          TV     EQU $A656
0800          ;
0010          DATA   ORG $0010
0010          ;
0010 50              BYT $50            ;R  THESE ARE SEGMENT
0011 79              BYT $79           .;E  CODES FOR "READY"
0012 77              BYT $77            ;A
0013 5E              BYT $5E            ;D
0014 6E              BYT $6E            ;Y
0015 00              BYT $00            ;SPACE
0016          ;
0200          TIME1  ORG $0200
0200 20868B   START  JSR ACCESS
0203 F8              SED
0204 A900     RESET  LDA #$00           ;INITIALIZE FRACTION
0206 850A            STA FRACNT         ;COUNTER
0208 A500            LDA HRSSET
020A 8507            STA HRSREM         ;TRANSFER TIME SET
020C A501            LDA MINSET         ;VALUES TO TIME
020E 8508            STA MINREM         ;COUNT LOCATIONS
0210 A502            LDA SECSET
0212 8509            STA SECREM
0214 A904            LDA #$04           ;INITIALIZE MODE OF OPER-
0216 8D7902          STA STOMOD+1       ;ATION TO RESETABLE..
0219 78       CONTIN SEI                ;INHIBIT INTERRUPTS
021A A205            LDX #$05
021C B510     LODE   LDA $10,X          ;LOAD DISPLAY:BUFFER
021E 9D40A6          STA DISBUF,X       ;WITH "READY"...
0221 CA              DEX
0222 10F8            BPL LODE
```

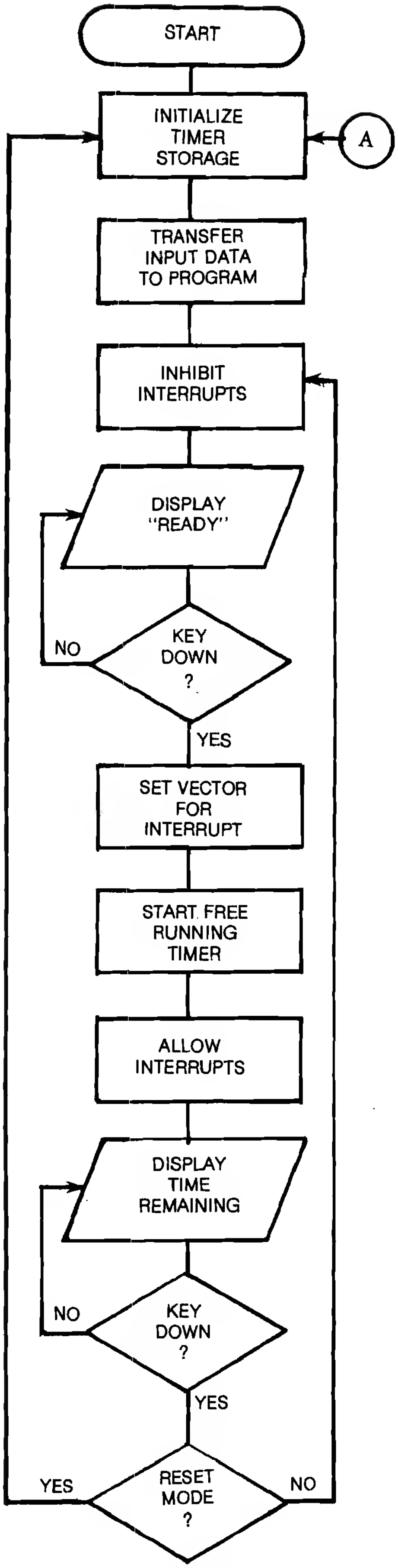


***Figure 1:* Main Routine**

```
0224 200689   SHORDY JSR SCAND         ;SHOW "READY"
0227 208683          JSR INSTAT        ;CHECK FOR KEY DOWN
022A 90F8            BCC SHORDY        ;IF KEY NOT DOWN, SHOW "READY"..
022C A900            LDA #$00
022E 8D7EA6          STA $A67E         ;LOAD IRQ VECTOR
0231 A903            LDA #$03          ;WITH ADDRESS 0300
0233 8D7FA6          STA $A67F
0236 A9C0            LDA #$C0          ;LOAD IER VECTOR
0238 8D0EA0          STA $A00E
023B AD0DA0          LDA $A00D         ;LOAD IFR REGISTER
023E 29BF            AND #$BF
0240 8D0DA0          STA $A00D
0243 A9C0            LDA #$C0          ;SET TIMER FOR
0245 8D0BA0          STA $A00B         ;FREE RUN MODE,
0248 A950            LDA #$50          ;THEN..
024A 8D06A0          STA $A006         ;LOAD HIGH & LDW ORDER
024D A9C3            LDA #$C3          ;COUNTER LATCHES TO
024F 8D05A0          STA $A005         ;GIVE 50MS INTERRUPT
0252 58              CLI
0253 A507     LOAD   LDA HRSREM        ;GET
0255 850B            STA DISPIN        ;AND
0257 206E03          JSR LODISP        ;LOAD
025A A508            LDA MINREM        ;TIME
025C 850B            STA DISPIN        ;REMAINING
025E 206E03          JSR LODISP        ;INTO DISPLAY 8UFFER
0261 A509            LDA SECREM        ;HRS.
0263 850B            STA DISPIN        ;MIN.
0265 206E03          JSR LODISP        ;SEC.
0268 A9AC            LDA #$AC          ;DETERMINE MODE SELECTED
026A C506            CMP MODE          ;AND
026C F00D            BEQ SETMOD        ;CHANGE IF REQUIRED.
026E A904     SHOW   LDA #$04          ;DISPLAY TIME REMAINING
0270 8D56A6          STA TV            ;UNTIL INTERRUPTED
0273 205A83          JSR DELAY         ;OR KEY IS DOWN
0276 90D8            8CC LOAD
0278 4C0402   STOMOD JMP RESET         ;04 MAY BE CHANGED TO 19
027B A919     SETMOD LDA #$19          ;UNDER PROGRAM CONTRDL,
027D 8D7902          STA STOMOD+1      ;DEPENDS ON MODE.
0280 4C6E02          JMP SHOW
0283 207289   WARN   JSR 8EEP          ;BEEPER RDUTINE
0286 207289          JSR BEEP          ;FOR WARNING
0289 207289          JSR BEEP
028C 207289          JSR BEEP
028F 207289          JSR BEEP
0292 60              RTS
0293 207289   TIMOUT JSR 8EEP          ;BEEP!
0296          ;
0296 208683          JSR INSTAT        ;CHECK FOR KEY DOWN
0299 90F8            BCC TIMOUT        ;IF NO KEY DOWN, BEEP AGAIN
029B 4C0002          JMP START         ;IF KEY DOWN, JUMP TO START
029E          ;
029E          ;
0300          TIMER  ORG $0300
0300 08              PHP               ;INTERRUPT RDUTINE
0301 48              PHA               ;STARTS HERE SO
0302 8A              TXA               ;SAVE IMPORTANT
0303 48              PHA               ;REGISTERS
0304 98              TYA
0305 48              PHA
0306 E60A            INC FRACNT        ;INCREMENT FRACTIONS
0308 A50A            LDA FRACNT        ;OF A SECOND COUNTER
030A C914            CMP #$14          ;IF FULL SECOND IS
030C F003            BEQ REFRAC        ;NOT UP YET JUMP TD
030E 4C6403          JMP ENDINT        ;END INTERRUPT ROUTINE
0311 A900     REFRAC LDA #$00
0313 850A            STA FRACNT        ;AFTER RESETTING FRACTIDN
0315 38              SEC               ;COUNTER SUBTRACT
0316 A509            LDA SECREM        ;ONE SECOND FROM TIME
0318 E901            SBC #$01          ;REMAINING. IF NOT TIME
031A 8509            STA SECREM        ;TO SUBTRACT 1 MINUTE
031C C999            CMP #$99          ;GOTO "CHKEND"
031E D01A            8NE CHKEND        ;ROUTINE
0320 A959            LDA #$59          ;ONE MINUTE HAS ELAPSED
0322 8509            STA SECREM        ;SO RESET SECONDS TO 59
0324 38              SEC
0325 A508            LDA MINREM        ;SUBTRACT 1 MINUTE
0327 E901            SBC #$01          ;TIME REMAINING
0329 8508            STA MINREM        ;IF NOT TIME TO SUBTRACT
032B C999            CMP #$99          ;ONE HOUR....
032D D00B            BNE CHKEND        ;GOTO "CHKEND" ROUTINE
032F A959            LDA #$59          ;ONE HOUR HAS ELAPSED
0331 8508            STA MINREM        ;SO RESET MINUTES TO 59
0333 38              SEC ·
0334 A507            LDA HRSREM
0336 E901            S8C #$01          ;SUBTRACT 1 HOUR FROM
0338 8507            STA HRSREM        ;TIME REMAINING
033A A507     CHKEND LDA HRSREM        ;IF HRS., MIN. AND
033C C900            CMP #$00          ;SEC. ARE ALL ZERO
033E D00F            BNE CHECK         ;THEN TIMER HAS TIMED OUT
0340 A508            LDA MINREM
0342 C900            CMP #$00
0344 D009            BNE CHECK         ;IF NO TIME OUT
0346 A509            LDA SECREM        ;HAS DCCURRED GO TO
0348 C900            CMP #$00          ; CHECK AND DETERMINE
034A D003            BNE CHECK         ;IF TIME FOR WARNING
034C 4C9302          JMP TIMOUT        ;IS TD BE SOUNDED
034F A505     CHECK  LDA SECWAR        ;COMPARE WARNING HR/MIN/SEC
```

```
0351 C509           CMP SECREM      ;TO TIME REMAINING HR/MIN/SEC
0353 D00F           BNE ENDINT      ;IF NOT A MATCH, GOTO
0355 A504           LDA MINWAR      ;END INTERRUPT ROUTINE
0357 C508           CMP MINREM
0359 D009           BNE ENDINT
035B A503           LDA HRSWAR
035D C507           CMP HRSREM
035F D003           BNE ENDINT
0361 20B302         JSR WARN        ;SOUND A WARNING
0364 AD04A0  ENDINT LDA $A004       ;RESET TIMER INTERRUPT FLAG
0367 68             PLA             ;RESTORE
0368 A8             TAY             ;ALL
0369 68             PLA             ;PREVIOUSLY
036A AA             TAX             ;SAVED
036B 68             PLA             ;REGISTERS
036C 28             PLP             ;AND
036D 40             RTI             ;RETURN FROM INTERRUPT...
036E A50B    LODISP LDA DISPIN      ;GET DIGITS TO BE
0370 850C           STA DISPOP      ;DISPLAYED AND SAVE
0372 6A             ROR             ;FOR LATER RECALL.
0373 6A             ROR             ;POSITION MSD FOR CONVERSION
0374 6A             ROR             ;TO ASCII
0375 6A             ROR
0376 200983         JSR NIBASC
0379 20C189         JSR OUTDSP      ;LOAD DIGIT
037C A50C           LDA DISPOP      ;GET OTHER DIGIT AND CONVERT
037E 200983         JSR NIBASC      ;TO ASCII....
0381 20C189         JSR OUTDSP      ;THEN LOAD DIGIT
0384 60             RTS             ;RETURN FROM SUBROUTINE....
```

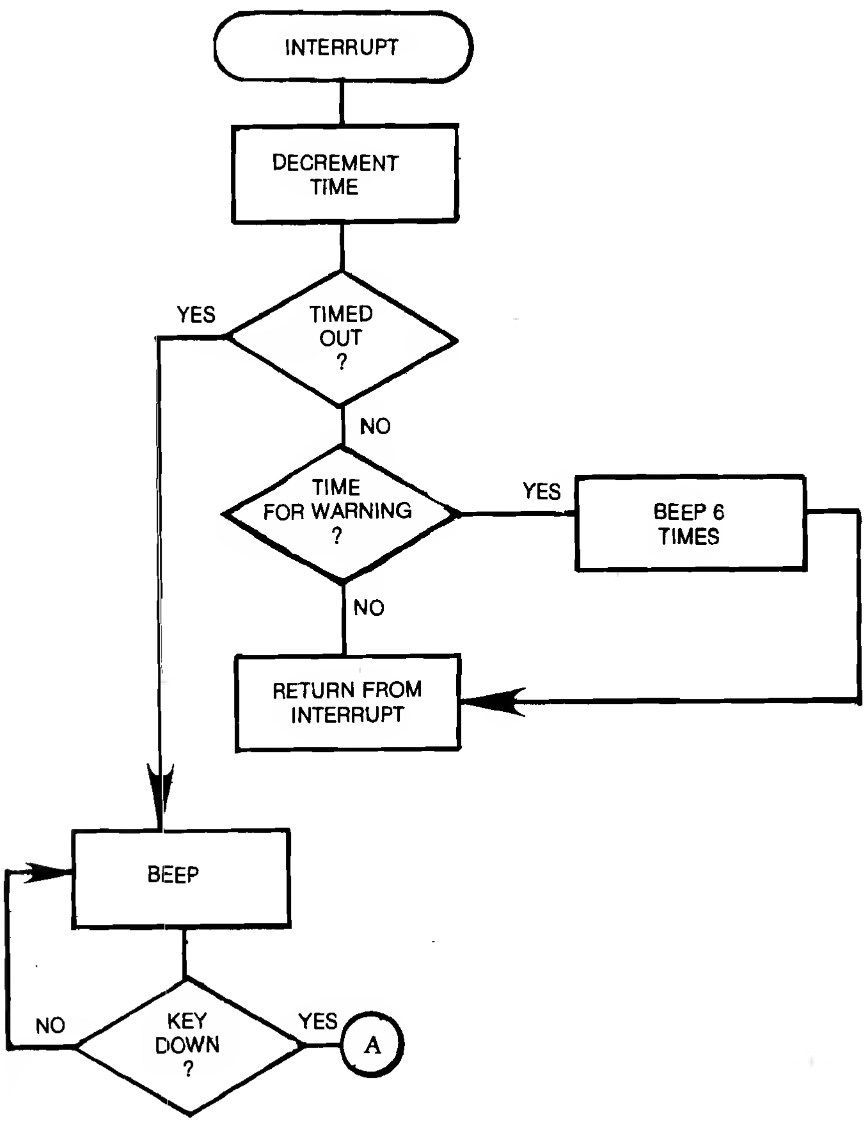


***Figure 2:* Interrupt Routine**

in such an effort. So, you guessed it, here comes the SYM Timer. Starting at address "0000" you punch in "10" "30" "00". Then to provide a one-hour warning you continue with "01" "00" "00". At this point you are ready to select mode of operation.

By entering "AC" you will select the "Accumulative" mode of operation. In this mode you can "start" and "stop" the timer as often as required. The timer will continue timing at each "start" from where it was last stopped. If you had entered anything other than "AC" you would have selected the "Resettable" mode of operation. In this mode each "start" causes the timer to begin again from the original timer interval you set.

With a simple "Go" "200" "CR" SYM displays "ready". To start the timer press any key and time remaining is displayed. To stop the timer once more press any key and "ready" is displayed again.

When the timer reaches the pre-time out warning the beeper will sound momentarily, and when time out occurs, the beeper will sound continuously until it is reset by pressing any key.

Well that's it—maybe! I keep fighting off the urge to toss in more and more. For instance, how about a 1 year timer that reads out "Hrs × 100" "Hrs." "Min."? Or if that's a little ridiculous, then maybe one that displays "days" "hrs." "min." for in excess of 3 months of timing. Then of course we could have an option to display elapsed time as well as time remaining. I don't suppose it would be too difficult to toss in a 24 hour clock while we're at it. Of course it would have to operate simultaneously with all the other options.

So on and on it goes. For now, however, I will leave it to others to perfect the *ultimate* time machine.

MICRO

GET FREE ADS ON TV!

How? Use our ADVERTISING SOFTWARE! You put it in the APPLE and produce colorful, dynamic ads on the screens of TV sets in your shop window. Even if you are not a shop owner, you can use this software to broadcast messages on TV screens in schools, hospitals, factories, etc. The following message-making programs are available.

SUPER MESSAGE: Creates messages in full-page "chunks". Each name allows statements of mixed typestyles, typesizes and colors, in mixed upper & lower case. Five typestyles are available. They range from regular APPLE characters, up to double-size, double-width characters with a heavy, bold font. Six colors may be used for each different typestyle. Vertical & horizontal centering are available, and word-wrap is automatic. Users can chain pages together to make multi-page messages. Pages can be advanced manually or automatically. Multi-page messages can be stored to disc or recalled instantly.
REQUIRES 48K & ROM APPLESOFT .... $ 50.

MULTI-MESSAGE with INTERLEAVED COLOR PATTERNS: Up to 10 messages can be run in sequence. Colorful, dynamic patterns (kaleidoscope or abstract art) can be interleaved between messages, at user option. Consists of 28 crisp, readable characters/lineX4 lines/pageX3 pages of text per message. Characters are 1/8 screen-height and "puff" onto the screen at comfortable reading speed.
REQUIRES 32K & INTEGER BASIC.......................... $ 35.

HI-RES ALPHANUMERIC MESSAGE: Same as Multi-Message above, but has only one message/set and no interleaved color-patterns. Still very good general message-maker!

THE SCROLLING WONDER: 4 brief messages appear in APPLE uppercase characters by "floating" onto the screen from below. Messages enter in random sequence, with random 50% of messages "flash". A multiple-rainbow grand finale ends the program. Very good program to run at point of purchase.

GIANT LETTER: Brilliantly-colored letters, of full screen height, appear one-at-a-time, in sequence, to spell out messages. Successive words have different colors. A running summary of letters, in APPLE characters, appears in the bottom 4 lines of the screen, as the giant letters are presented. Very good program for shop windows.
ALL 3 ABOVE TOGETHER, ON DISK, FOR 32K, INTEGER BASIC .... $ 30.

LET APPLE PLOT YOUR DATA AND KEEP YOUR RECORDS TOO!

APPLE DATA GRAPH 2.1: Plots up to 3 superimposed curves on the Hi-res Screen both the X & Y axes dimensioned. Each curve consists of up to 120 pieces of data. Graphs can be stored to disc and recalled immediately for updating. Up to 100 graphs can be stored on the same disc. Great for Stock-market Charting, Business Management, and Classroom Instruction!
REQUIRES 48K & ROM APPLESOFT .... $ 40.

APPLE RECORD MANAGER: Allows complete files to be brought into memory so that record searches and manipulations are instantaneous. Records within any file can contain up to 20 fields, with user-defined headings. Information can be string or numeric. Users can browse thru files using page-forward, page-backward or random-search commands. Records can easily be searched, altered or sorted at will. Files can be stored on the same drive as the master program, or on another, if a second drive is available. Records or files can be printed, if desired. Additional modules coming are a STATISTICS INTERFACE, CHECKBOOK, MAILING LIST & DATA-ENTRY.
REQUIRES 48K & ROM APPLESOFT .... $ 35.

* All Software above on Disk for APPLE DOS 3.2

GET FREE SOFTWARE TOO!

How? Order any of the items below, and for each $100 worth of merchandise ordered, we will give you one of the items at left for FREE!

APPLE ADD-ONS

HAYES MICROMODEM for APPLE	$ 300.
Z80 SOFTCARD by MICROSOFT	$ 275.
16K RAMCARD by MICROSOFT	$ 159.
FORTRAN for APPLE by MICROSOFT	$ 159.
COBOL for APPLE by MICROSOFT	$ 599.
BASIC Compiler for APPLE by MICROSOFT	$ 315.

PRINTERS

CENTRONICS 737 (3 mo. warranty)	$ 795.
CENTRONICS 737 (15 mo. warranty)	$ 915.
EPSON TX-80 with TRACTORS & GRAPHICS	$ 645.
EPSON MX-80 with TRACTORS & 132 Columns	$ 515.
PAPER TIGER 460G with GRAPHICS & 2K Buffer	$ 1135.
PAPER TIGER 445G with GRAPHICS & 2K Buffer	$ 749.

WORD PROCESSING

EZ WRITER PROFESSIONAL SYSTEM for APPLE	$ 239.
EZ MAILER (Interfaces to EZ WRITER above)	$ 65.
VIDEX VIDEOTERM (80-Column Card for APPLE)	$ 295.
VIDEX VIDEOTERM (Same as above with GRAPHICS)	$ 320.
SUP'R'TERM (80-Column Card for APPLE)	$ 320.

BUSINESS PROGRAMS for APPLE & TR-80 by SPECTRUM SOFTWARE

MICROACCOUNTANT: An ideal package for the very small business, based upon classic T-accounts & Double-Entry Bookkeeping. This efficient program records and produces reports on account balances, general ledger journals, revenues & expenses. 40-column or screen reports. Handles up to 1000 journal entries/month, for up to 300 accounts. Includes a short primer in Financial Accounting.
REQUIRES 48K & ROM APPLESOFT .... $ 49.

BUSINESS CHECK-REGISTER with BUDGET: Unique system allows setting up predefined purpose & recipient accounts (50 each). Supports unique names too. Rapid-access to check-files with scrolling display & 40-col. printout, if desired. Up to 100 checks/mo. + reconciliation + AUTOMATIC BUDGET VARIANCES!
REQUIRES 48K & APPLESOFT ROM .... $ 49.

STOCK MARKET

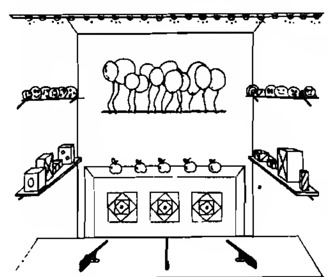
STOCK MARKET ANALYSIS for APPLE by GALAXY .... $ 49.

COD'S & Personal Checks are Welcome!

CONNECTICUT INFORMATION SYSTEMS CO.
218 Huntington Road, Bridgeport, CT 06608 (203) 579-0472

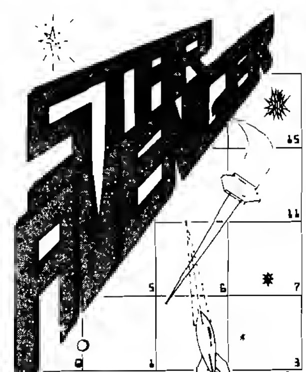
Asteron - The definitive hi-res implementation of the Asteroids arcade game. Features: Ship movement, hyper-space, alien saucers, sound effects, graphic routines allowing up to 25 objects to be displayed with real time response. Played from paddles or keyboard. **$27.50**

APPLE ARCADE



Shooting Gallery

Shooting Gallery - A real time simulation of a midway arcade. Features: row targets, pop targets, different skill levels, and bonus time. May be played using either game paddles or joy sticks. **$22.50**



Star Avenger -High speed guerrilla warfare in space pitting you against your Apple. Featuring a new universe each game and varying skill levels. Universe consists of 16 hi-res screens with instantaneous crossover. **$27.50**

Available Now!

MultiBoot™ Upgrade

Have you not wished that your Basic software would work in both DOS 3.2 *and* DOS 3.3? Tired of spending hours "Muffin"ing your old programmes? Now your problems are solved with MultiBoot™ Upgrade. Upgrade a whole disc in just seconds and use your disc on any of DOS 3.3, DOS 3.2 and the Language Card.
$50.00

All Western MicroData game software is written in assembly language for maximum speed. All programmes require 48K and DISK DRIVE and will work on standard Apple II, Apple II plus, and Pascal systems, with either DOS 3.2 or DOS 3.3.

Western MicroData Enterprises Ltd.
P.O. Box G33, Postal Station G
Calgary, Alberta
Canada T3A 2W1
1-403-247-1621

For U.S. and foreign orders, prices are in U.S. dollars. For Canadian orders, prices are in Canadian dollars. Send cheque or Postal Money Order only. Allow 3-4 weeks for cheque to clear if not certified and allow up to 4 weeks for delivery.

Dealer and Computer Club enquiries invited.

Apple is a registered trademark of Apple Computer Inc.
Disk II is a registered trademark of Apple Computer Inc.

Oh No — It's Garbage Collect!

This article describes Garbage Collect in Microsoft's 6502 BASIC. The worst case is described, and a few suggestions made on how to avoid it.

Gordon A. Campbell
36 Doubletree Road
Willowdale, Ontario
M2J 3Z4

I was really pleased! The simple text editor worked well. It even had a fancy quasi-INPUT routine, just like CURSOR. And it only took a couple of days to develop, since it was written entirely in BASIC. Now to get on with some articles.

The first opus went well. After several input sessions, I ran a full draft. All the changes were going well, when suddenly, right in the middle of entering a command, the PET went dead. Pushing the STOP key did nothing, so I sat back to consider my sins. After a minute, the cursor returned, and the editor was again working.

The light came on. I was the victim of the dreaded garbage collect.

Garbage collect is the compression of string space. In Microsoft's 6502 BASIC, string contents are placed at the top of memory, working down. When a string is assigned a new value, it is placed below all previous strings. At some point, memory is filled, so garbage collect squishes all the strings back up to the top. It may also be forced, by using FRE(0).

The following one-liner provides the basis for some experiments:

```
5 N=1000:
  B$=CHR$(1):
  DIM A$(N):
  FOR J = 0 TO N:
  A$(J) = B$:
  NEXT:
  A$(0) = "B":
  T=TI:
  PRINT FRE(0) TI-T
```

The program sets up bunches of strings, changes the first one, and forces garbage collect while printing the time required. (The CHR$ is required since assigning a string a literal value results in the string pointer pointing at the literal in the program, rather than use of string space.)

Changing N showed that the number of strings has a roughly exponential effect on the time required. Changing the size of B$ showed that the number of characters in the strings has no apparent effect on the time.

To find the worst case, some swift calculation shows that N can be set to 7908, and garbage collect takes 84 minutes and 13 seconds. But we can go higher. Drop the start-of-BASIC down to the first cassette buffer, and raise the top-of-memory to the end of the screen. Now N can be set to 8261, for a time of 91 minutes and 56 seconds!

How about other machines? A call to a friend showed that Applesoft is compatible with the PET. The only difference is the 10% that the PET spends looking at the keyboard and cassettes, and updating the clock. Down at our neighborhood Radio Shack we found that string space must be reserved with a 'CLEAR n' command. There is no apparent time spent in garbage collect, but there is a value for the CLEAR command which seems to crash the system, so that may be it. Presumably the Atari with its fixed-length strings doesn't create garbage in the first place.

Published information indicates that the latest PET ROM-set does garbage collect much more quickly. In a classic trade-off of speed versus memory, it also takes two bytes more per string. The ways to reduce garbage collect are fairly obvious: don't have more strings than are absolutely required. For example:

1. Re-use work variables.
2. Use numbers rather than strings for switches.
3. Put literals right into PRINT statements rather than use constant strings.
4. Try to create the most stable strings first.
5. Avoid loops which create a string by concatenating a character at a time onto the string.
6. Apply the usual techniques to keeping your program small.
7. Avoid sorting techniques which involve changing the actual contents of the array. Instead, use QUICKSORT, or an Assembler sort which changes the string pointers.

Garbage collect will happen in any case. In interactive programs without a large number of strings, it can be made invisible to the operator by forcing it (X=FRE(0)) during times when the operator doesn't expect to use the keyboard. For 'batch' programs, the least amount of time will be consumed by just letting it happen when it must.

In summary, the next time your PET (Apple, SYM) seems to crash, don't reset it right away. It may just be collecting its garbage.

After 15 years in main-frame data processing, Gordon Campbell purchased a cassette-based PET in the spring of 1979. Since then, the PET has grown a disk, printer, and modem. The latest expansion provides CB2 sound in stereo.

MICRO

An AIM 65 Intelligence Test

Please answer each question:

1 Who has been offering complete 6502-based products since 1976?
☐ Rockwell International ☐ The Computerist

2 Who makes a video expansion board for the AIM 65 that fully supports the AIM Monitor, Editor, Assembler and BASIC?
☐ Rockwell International ☐ The Computerist

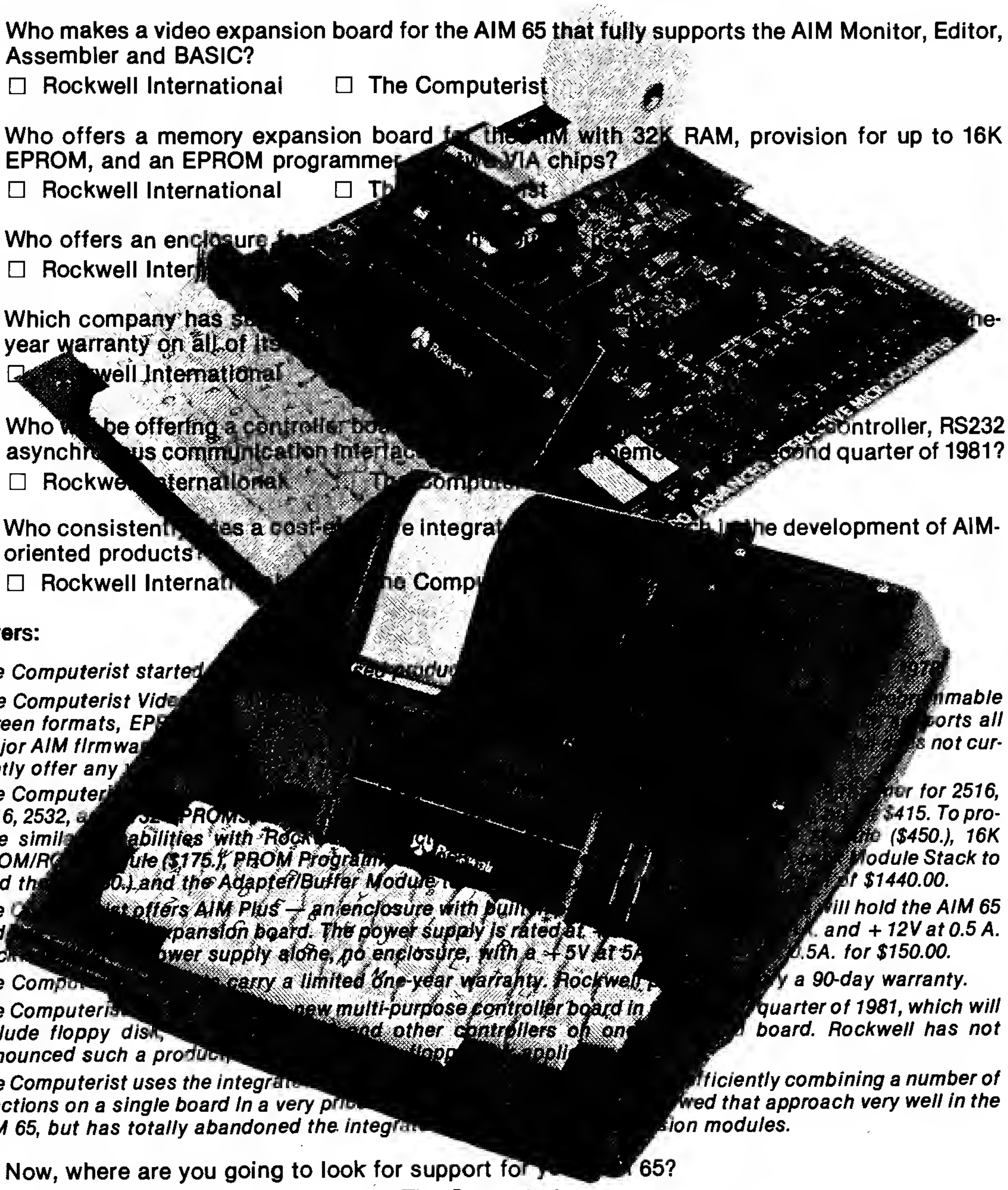
3 Who offers a memory expansion board f... with 32K RAM, provision for up to 16K EPROM, and an EPROM programmer ... VIA chips?
☐ Rockwell International ☐ Th...

4 Who offers an enclosure f...
☐ Rockwell Inter...

5 Which company has s... one-year warranty on all of its...
☐ ...well International

6 Who w... be offering a controller bo... controller, RS232 asynchr...us communication interfa... mem... ...ond quarter of 1981?
☐ Rockwe... ternational ☐ The Computer...

7 Who consistent... ...es a cost-e... ...e integrat... ...the development of AIM-oriented products?
☐ Rockwell Internat... ☐ ...he Comp...



Answers:

1. *The Computerist started ... products ... 1976.*
2. *The Computerist Vide... ...mmable screen formats, EP... ...ports all major AIM firmwa... ...s not currently offer any ...*
3. *The Computer... ...er for 2516, 2716, 2532, a... ...PROMs... $415. To provide simil... ...abilities with Rock... ...le ($450.), 16K PROM/R... ...ule ($175.), PROM Program... ...Module Stack to hold th... ...0.) and the Adapter/Buffer Module ... ...f $1440.00.*
4. *The C... ...st offers AIM Plus — an enclosure with buil... ...will hold the AIM 65 and ... ...pansion board. The power supply is rated at ... and + 12V at 0.5 A. Roc... ...wer supply alone, no enclosure, with a + 5V at 5A ... 0.5A. for $150.00.*
5. *The Compu... ...carry a limited one-year warranty. Rockwell ... ...y a 90-day warranty.*
6. *The Computeri... ...new multi-purpose controller board in ... quarter of 1981, which will include floppy dis... ...and other controllers on on... board. Rockwell has not announced such a produc... ...flopp... ...ppli...*
7. *The Computerist uses the integra... ...fficiently combining a number of functions on a single board in a very pri... ...wed that approach very well in the AIM 65, but has totally abandoned the integra... ...sion modules.*

8 Now, where are you going to look for support fo... ...65?
☐ Rockwell International ☐ The Computerist

Answer: *If you answered "The Computerist" to question 8, then you pass the AIM Intelligence Test. So send for our 1981 Product Guide. If you answered "Rockwell International", then maybe you didn't understand the question.*

Prices listed are US only and do not include Shipping and Handling.

34 Chelmsford St., Chelmsford, MA. 01824 617/256-3649

THE COMPUTERIST®

MICRO

New Publications

Mike Rowe
New Publications
P.O. Box 6502
Chelmsford, MA 01824

This column lists new publications received for review and also reports on pertinent publication announcements received from book and periodical publishers.

General 6502

The 6502 Instruction Handbook by Scelbi Publications (20 Hurlbut Street, Elmwood, Connecticut 06110), 1981, 44 pages, 3¾ × 8½ inches, paperbound. $4.95

Designed as a shirt-pocket guide for programmers, technicians, and engineers. Portions of the publication appeared originally in *SCELBI's 6502 Software Gourmet Guide & Cookbook* (by Robert Findley, 1979). This slim reference work, available from computer stores or for an extra 50¢ from the publisher, contains a synopsis of each instruction set for the 6502 CPU. Mnemonics and machine codes in hexadecimal format are provided for each addressing mode. Appendices list the instruction set alphabetically by assembler mnemonics as well as numerically by machine code. Other information provided includes a hexadecimal-to-decimal conversion chart, a chip pinout diagram, timing data, and diagrams of chip architecture.

6502 Games by Rodney Zaks. 6502 Series, Volume IV, Sybex Inc. (2344 Sixth Street, Berkeley, California 94710), 1980, x, 292 pages, 50 figures, 5½ × 8½ inches, paperbound.
ISBN: 0-89588-023-9 $12.95

This book is designed as an educational text for the programmer who wants to learn advanced programming techniques by using the 6502. Although it can be used merely to play games with a 6502-based board, for educational purposes, the reader should be familiar both with the 6502 instruction set and with basic programming techniques. The programs listed are for the SYM but can be adapted to other 6502-based microcomputers.

CONTENTS: *Introduction*—The Games Board. *Music Player*—Play a sequence of up to 255 notes (13 different notes) and record it automatically. *Translate*—The computer displays a binary number. Each player in turn must press the hexadecimal equivalent as quickly as possible. The first to score 10 wins. Designed for two players. *Hexguess*—Guess a 2-digit hex number generated by the computer. The computer will tell you how far off your guess is. You are allowed up to 10 guesses. *Magic Square*—Light up a perfect square on the board. Each key inverts some LED pattern. Skill and logic are required. *Spinner*—A light is spinning around a square. You must catch it by hitting the corresponding key. Every time you succeed, it will spin faster. A game of skill. *Slot Machine*—A Las Vegas type slot machine is simulated, with three spinning wheels. Try your luck. *Echo*—Recognize and duplicate a sound/light sequence (also known as SIMON—A manufacturer trademark). *Mindbender*—Play against the dealer (the computer) with a deck of 10 cards. You may hit or stay. Don't bust! *Blackjack*—Guess a sequence of numbers generated by the computer. It will tell you how many digits are correct and in the right position (also known as MASTERMIND—a manufacturer trademark). *Tic-Tac-Toe*—Try to achieve three in a row before the computer does in this favorite game of strategy. The computer's ability improves with yours. Can you outsmart it? *Appendices: A.* 6502 Instructions—Alphabetic; *B.* 6502—Instruction Set: Hex and Timing. *Index.*

AIM 65

AIM 65 Laboratory Manual and Study Guide by Leo J. Scanlon. John Wiley & Sons (605 Third Avenue, New York, New York 10158) 1981, 180 pages, diagrams and charts, 8½ × 11 inches, paperbound.
ISBN: 0-471-06488-2 $7.95

A study and exercise book designed to introduce students to microcomputers by working with the AIM 65. Pages are perforated so that the student's answers, written in the book, can be handed in, lesson by lesson, for review by the instructor. The author, employed by Rockwell International, the manufacturer of the AIM 65, provides 32 pages of answers to the experiments.

CONTENTS: Getting to Know the AIM 65; Addition Operations; Subtraction and Logical Operations; Program Sequencing; Debugging Programs; Multiplication Operations, with Shift & Rotate; Division Operations; Subroutines and the Stack; Unordered Lists; Sorting Unordered Data; Code Conversion from Input; Code Conversion for Output; Input/Output; A More Powerful I/O Device, the R6522 VIA; Interrupts; A Timing Program with Decimal Output; The AIM 65 Assembler; Answers to Experiments.

General Microcomputer

The Personal Computer Book by Robin Bradbeer. Input Two-Nine, an imprint of MCB Publications Limited (198/200 Keighley Road, Bradford, West Yorkshire, England BD9 4JQ), 1980, 220 pages, illustrated, 8¼ × 5-6/8 inches, paperbound.
ISBN: 0-905897-56-0 U.S. $15.00; £5.25

An introductory work on microcomputers, written especially for readers in the United Kingdom.

CONTENTS: *What's It All About?*—The computer can assist us tremendously, both in business and pleasure; How is it possible?; The first hobby computer; Who buys personal computers?; What do you use the computer for?; Developments in the next few years. *Where Do I Start?*—Ten hints to help you on your way. *The Computer—What Is It? How Does It Work?*—The computer—confusingly versatile; How the computer works, in simple terms; Binary numbers; How does the computer handle binary numbers?; The processor —the CPU—from the inside; The computer's own road network—the bus; Storage inside the computer. *How Do I Talk to the Computer?*—Machine Language; Assembly language; High-level languages; At which level do I begin?; BASIC—a convenient language; Firmware; Software; Which microprocessor is best? *What's In the Boxes?*; Input devices; Keyboard-based input; Speech recognition; Direct Input; Storage media; Cassette storage; Disk storage; Other storage media; Output devices; Video output; Printed output; Electric typewriter/TTY; Matrix printers; Daisy wheel printer; Other printers; Speech synthesis. *What Can I Buy?*—The Computer system; Personal computer equipment survey; Part 1, Section A—*Single board computers;* Kit-built systems; Training systems, Part 1, Section B—*Desk top systems.* Part 1, Section C—*Bus-based systems*—S100 Bus. Part 1, Section D—*Other Buses*—SS 50, Non-standard. Input/Output devices, memory storage media; other media; Part 2—*Printers.* Part 3—*Video display units.* Part 4—*Other peripherals.* How do I choose a system? *What Can I Do With It?*—Games; Education; Business use; Word processing; Information handling; Controlling things; Making money; Examples of personal computers in use. *Appendices: A.* Binary Arithmetic; Octal; Hexadecimal; ASCII Code. *B.* Bus Standards; S100 (IEEE); SS50, etc. *C.* Manufacturers and Distributors in U.K. *D.* Computer Clubs in the U.K. *E.* Magazines in English... UK/USA/Continent. *F.* Bibliography of Selected Microcomputer Books. *G.* Glossary. *H.* Some Hints on Kit-build Systems.

(Continued on following page)

The Carl Helmers Personal Computer Letter is a monthly newsletter which began publication with the January 1981 issue. Helmers, a co-founder of *Byte* magazine and its former Editorial Director, provides subscribers with analyses of issues and trends affecting the small computer industry. Helmers plans to offer subscribers the opportunity to participate in a monthly Personal Computer Industry Conference Call which he will moderate. Each issue is a minimum of 8 pages; some may run to 24 or 48 pages. A one-year subscription is $200.00 from North American Technology, Inc., 174 Concord Street, Suite 23, Peterborough, New Hampshire 03458.

Microcomputers and Business

Basic Business Software by E.G. Brooner. Blacksburg Continuing Education Series, Howard W. Sams & Co., Inc. (4300 West 62nd Street, Indianapolis, Indiana 46268), 1980, 142 pages, charts, diagrams, and listings, 5½ × 8½ inches, paperbound.
ISBN: 0-672-21751-1 $9.95

This book is designed primarily for business people who want to understand some of the fundamentals of business software development. But it is also for programmers who want to learn more about business software. Some familiarity with BASIC-language programming is assumed. The author aims to teach readers either to write some of their own business software or evaluate programs written by others. Sample programs are included.

CONTENTS: *Introduction to Small-Business Software*—Objectives; Small-Business Computers Defined; Effect on Paper Work; Businesses that Benefit; Software Costs; Self-Help Test Questions. *Software Fundamentals*—Objectives; Software Functions; Computer Languages; The Operating System; Software Defined; Language-Independent Programming; BASIC Comparison and Translation; Self-Help Test Questions; References. *How To Choose Appropriate Business Software*—Objectives; Practical Limits; Where To Get It; When Customizing Is Needed; Compatability; Self-Help Questions. *How Programs Are Put Together*—Objectives; Terms Defined; The Use of Symbols in Programming; The Step-by-Step Method; Subroutines, or Modules; Programming Hints; Summary of the Step-by-Step Method; Debugging Hints and Other Techniques; The Disk Subsystem; Self-Help Test Questions. *Information Storage and Retrieval*—Objectives; Data Generation and Storage; How Data Is Stored; Disk Handling; The Disk Library; File Structure; Overview of "CHECKING" Program; Sorting Computer Data; Program Analysis; "NAMELIST"; Self-Help Test Questions; References. *Inventory Control*—Objectives; Who Needs an Inventory?; Inventory as a List; Inventory Functions; Program Analysis; Program Evaluation; Rapid Search Methods; Summary; Self-Help Test Questions; Reference. *Payroll Programs*— Objectives; Payroll Requirements; Printing on Prepared Forms; Program Development; The Master File; Master Payroll Program; Entering Employee Data; Payroll Entries and Calculations; Entering Pay Data; Pay Procedure; Payroll Summary; Conclusion; Self-Help Test Questions; Test Programming Project. *General Ledger Programs*—Objectives; Terms Defined; General Ledger Defined; System Overview; Transaction Examples; Program Descriptions; Operating the General Ledger System; Self-Help Test Questions; Test Project. *An Introduction to Word Processing*—Objectives; Word-Processing Functions; Office of the Future; Hardware Requirements; Suitable Software; Time Sharing. *Basic Computer Modeling and Simulation*—Objectives; The Break-Even Example; Graphical Method; The Computer Technique; Other Simulation Problems; Random Numbers; Self-Help Test Questions; References. *Appendix*—ASCII Code Chart; Glossary; Index.

Small Computers for the Small Businessman by Nicholas Rosa and Sharon Rosa, dilithium Press (30 N.W. 23rd Place, Portland, Oregon 97210), 1980, x, 332 pages, 5½ × 8-5/16 inches, paperbound.
ISBN: 0-918398-31-2 $12.95

This book is written for *small business* people and is mostly about small computers, specifically microcomputers. It is intended to help the reader select the computer that best meets his business needs.

CONTENTS: *The Small Computer Revolution*—But we're not trying to sell you; "The price of a new car"; Then why shouldn't you wait?; Now, about that rash idea...; Affording it; Turnkey in the store; Graphics; How "big" a system; Making money directly; "But I'm not a computer freak..."; That mini- and micro- distinction; What about just renting services?; Now whaddaya mean, "Revolution?"; The integrated circuit; But what's a semiconductor?; Large scale integration; And suddenly—; Voila!; The significance. *The Small Business Computer*—Interfacing; Memories are made of this...; A final memory; Mass storage; The other stuff. *This Thing Called Software*—Documentation; Programs; Computer languages; Those translating programs; What BASIC looks like; Enough, already; Now, about that problem...; Acquiring the stuff. *Data Processing and Word Processing*—The nature of data processing; The nature of word processing; Choosing a system. *How to Shape Your Computer System*—Getting into it; Using the consultant; Finding the consultant; Finding the vendor; Getting it all in writing; Involving your staff; The happy outcome. *Buying Services Instead*—Service bureaus; Timesharing; Whither timesharing? Amen, amen. *The Minicomputer*—But anyway; Acquisition notes; Again, what's a mini?; Making a decision; The cloudy crystal ball; The onrushing dawn. *Shopping for Your Hardware*—How much to buy?; The double system; System in one cabinet? Memory options; Where to buy; Guarantees; Notes on I/O devices; Keyboards; Writehander™; Teleprinters, Teletype™; Electric typewriters; CRT display; Other displays; Cassette drive; Floppy disks, diskettes; Hard disks: Winchesters; Printers; Isolators, noise suppressors; Power supplies; Front panel; Modems; The computer room; "Desk tops" and accessories; Cost and quality; Watch out. *The Professions and the Computer*—The accountant; The law office; The doctor's office; The writer's office; That bottom line (financing); Leasing; Tax benefits. *The Butcher, The Baker and The Candelstick Jobber*—The small manufacturer; Construction and lumber; Warehouses; Real estate; Insurance; A portrait studio; Pharmacy; Restaurant; Finding out more. *Glossary. Appendix*—How It All Works. *Index.*

General Computer

Software News — The Computer Software Products Newspaper is a newspaper tabloid which will appear monthly beginning in May. Sentry Database Publishing, a division of Technical Publishing, will issue the tabloid (Technical Publishing is the publisher of *Datamation* and is owned by The Dun & Bradstreet Corporation). *Software News* will report on the software industry. It will provide analysis and commentary on applications packages, systems software, productivity aids, databases, and language processors. It will cover data and software security, software legal issues, and job opportunities; and it will offer user ratings and surveys, software vendor profiles, market statistics, and other business and financial information. The newspaper will be distributed to 50,000 software buyers and specifiers. For information, write Software News, 5 Kane Industrial Drive, Hudson, Massachusetts 01749.

MICRO Dealers

Presented here in zip-code order are those MICRO dealers who responded to our newsletter request for information concerning their dealership. Many have been MICRO dealers for quite some time while others are new. This service is provided to acquaint readers with these dealers and to encourage readers to visit dealers in their area. This listing is provided twice a year to update previous listings (see MICRO 29:69). This is not intended as a complete listing of MICRO dealers.

United States

Maine

Maine Micro Systems Inc.
555 Center Street
Auburn, Maine 04256
Contact: Hugh Blair/Al Celetti
207/782-7139
Hardware: Aim, Apple, Atari, CBM, KIM, PET, TRS-80
Software: Educational, Personal, Business, Games
Publications: MICRO, Compute, Kilobaud, 80-Microcomputing

Vermont

Computers Plus
177 Church St.
Burlington, Vermont 05401
Contact: Tim Barden
802/658-5858
Hardware: Apple, Atari, HP-85
Software: Educational, Personal, Business, Games, General Accounting
Publications: MICRO, Byte, Creative Computing, Microcomputing

Connecticut

The Computer Store
63 South Main St.
Windsor Locks, Connecticut 06096
Contact: Susan Bramley
203/627-0188
Hardware: Apple, HP-85
Software: Educational, Personal, Business, Games
Publications: MICRO, Byte, Nibble, Creative Computing

Southbury Professional Systems Inc.
D.B.A. The Micro-Computer Store
Union Square
Southbury, Connecticut 06488
Contact: Marilyn or Joseph Osterman
203/264-2983
Hardware: Apple, Atari, Vector Graphic
Software: Educational, Personal, Business, Games, Professional

New Jersey

Software City
111 Grand Ave.
River Edge, New Jersey 07661

New York

Time Enterprise
8247 Genesee Road
Springville, New York 14141
Contact: Paul Zielinski
716/592-7665
Hardware: OSI
Software: For OSI

Pennsylvania

Computer Mail Order
501 E. Third St.
Williamsport, Pennsylvania 17701
Contact: Randy Gailit
717/323-7921
Hardware: Atari, CBM, PET
Software: Educational, Personal, Business, Games

Maryland

Computer Crossroads, Inc.
9143 G Red Branch Road
Columbia, Maryland 21045
Contact: Richard Simpson
301/730-5513
Hardware: Apple, Atari
Software: Educational, Personal, Business, Games
Publications: Many

Virginia

Computerland of Tysons Corner
8411 Old Courthouse Road
Vienna, Virginia 22180
Contact: Rich Doud
703/893-0424
Hardware: Apple, Atari, CBM, PET, North Star, Dynabyte, T.I., Cromemco
Software: Educational, Personal, Business, Games, Languages, Utilities, etc.
Publications: MICRO, Byte, Kilobaud, Personal Computing, Creative Computing, Nibble

Computer Center
2927 Virginia Beach Blvd.
Virginia Beach, Virginia 23452
Contact: Jeff Wilson
804/340-1977
Hardware: Apple, Atari
Software: Educational, Personal, Business, Games
Publications: MICRO, Byte, Creative Computing, Interface, Call-APPLE

North Carolina

ETC Corporation
P.O. Box G - OLD NC 42
Apex, North Carolina 27502
Contact: Jeff Butler
919/362-4200
Hardware: AIM, CBM, KIM, OSI, PET, Billings
Software: Educational, Personal, Business, Games, Scientific, Custom
Publications: MICRO, Byte, Microcomputing, Compute

Florida

Associated Information Systems
825 Osceola Drive
Rockledge, Florida 32955
Contact: D.R. Hendricks
305/632-1090
Hardware: OSI
Software: Educational, Personal, Business, Games, Custom Programming
Publications: MICRO

AMF Microcomputer Center, Inc.
11158 N. 30th St.
Tampa, Florida 33612
Hardware: Apple
Software: Educational, Personal, Business, Games
Publications: All major computer magazines

Ohio

Microage Computer Store
2591 Hamilton Road
Columbus, Ohio 43227
Contact: John W. Spencer
616/868-1550
Hardware: Apple, Atari, North Star, HP, TI, Archives, Altos, Ithica, Zenith
Software: Educational, Personal, Business, Games
Publications: All

Michigan

New Dimensions in Computing, Inc.
541 E. Grand River
East Lansing, Michigan 48823
Contact: Robert Gibbs
517/337-2880
Hardware: Atari, Exidx, Vector Graphic, Intersystems
Software: Educational, Personal, Business, Games
Publications: MICRO, Byte, Interface Age, Microcomputing, 80-Microcomputing, Compute, Creative Computing

Wisconsin

Byte Shop of Milwaukee
6019 W. Layton Avenue
Greenfield, Wisconsin 53220
Contact: Kathleen Preston
414/281-7004
Hardware: Apple, CBM, PET, North Star, APF
Software: Educational, Personal, Business, Games
Publications: MICRO, 68 Micro, Byte, Creative, Kilobaud, 80 Microcomputing, onComputing, Interface Age, Nibble, Apple Orchard, Compute, Softside, Personal Computing, etc.

PETTED micro systems
4265 W. Loomis Rd.
(P.O. Box 21851)
Milwaukee, Wisconsin 53221
(moving soon)
Contact: Theodore J. Polczynski
414/282-4181
Hardware: Atari, CBM, KIM, PET,
Software: Educational, Personal, Business, Games, TRS-80 Instant software and creative computing
Publications: MICRO, Compute, Kilobaud, Creative Computing

Minnesota

Personal Business Systems
4306 Upton Ave. So.
Minneapolis, Minnesota 55410
Contact: Mike Carlson
612/929-4120
Hardware: Apple
Software: Educational, Personal, Business, Games
Publications: MICRO, Apple Orchard, Creative Computing, Peelings, Nibble

Illinois

Data Domain of Schaumburg
1612 E. Algonquin Rd.
Schaumburg, Illinois 60195
Contact: Marilyn Clark, Steve Shendelman
312/397-8700
Hardware: Apple, Alpha Micro, Hewlett-Packard Calculators and Accessories
Software: Educational, Personal, Business, Games
Publications: Large selection

Farnsworth Computer Center
1891 N. Farnsworth Ave.
Aurora, Illinois 60505
Contact: Luke Snyder
312/851-3888
Hardware: Apple, Hewlett-Packard, HP-85 A
Software: Educational, Personal, Business, Games
Publications: MICRO, Apple Orchard, Call APPLE, Kilobaud, 80 Microcomputing, Byte, Creative Computing, Interface Age

Wallace Computers, Inc.
2619 N. University
Peoria, Illinois 61604
Contact: Ronald A. Wallace
309/685-7876
Hardware: Apple
Software: Educational, Personal, Business, Games
Publications: MICRO, Byte, Call APPLE, Nibble, Softalk

Main Street Computer Company
215 N. Main
Decatur, Illinois 62523
Contact: David B. Herriott
217/429-5505
Hardware: Apple
Software: Educational, Personal, Business, Games
Publications: MICRO, Byte, Creative Computing, Interface Age

Missouri

Back to Basic Computer Center
43 Cross Keys
Florissant, Missouri 63033
Contact: Fred or Jane Hahn
314/837-4495
Hardware: Atari, OSI, Vector Graphic
Software: Educational, Personal, Business, Games
Publications: MICRO, Compute, Byte, Kilobaud, Microcomputing, Creative Computing

Computer Mart
1904 E. Meadowmere
Springfield, Missouri 65804
Contact: Rob Lurvey
417/862-6500
Hardware: Apple, Atari, IMS
Software: Educational, Personal, Business, Games
Publications: Most monthlies and quarterlies at present

Nebraska

Computers West
7351 Pacific St.
Omaha, Nebraska 68114
Contact: Kevin J. Burke
402/391-3737
Hardware: Apple, Atari, Data General, Archieves
Software: Educational, Personal, Business, Games
Publications: MICRO, Interface Age

Utah

Computer Translation, Inc.
1455 S. State Street
Orem, Utah 84057
Hardware: Apple
Software: Educational, Personal, Business, Games
Publications: Byte, Creative Computing, Personal Computing

Nevada

Home Computers
1775 E. Tropicana #6
Las Vegas, Nevada 89109
Contact: Ike
702/798-1022
Hardware: AIM, Apple, Atari, CBM, PET, Bally
Software: Educational, Personal, Business, Games, Functional Programs
Publications: Wide variety of computer magazines

The 6502 Program Exchange
2920 Moana Lane
Reno, Nevada 89509
Contact: David or Don Marsh
702/784-4991
702/825-8413 (eves.)
Software: Educational, Personal, Games, High-level Languages and Utilities

California

Malibu Microcomputing
2391DA DeVille Way
Malibu, California 90265
Contact: T. Nalevanko
213/456-1137
Hardware: Apple
Software: Educational, Personal, Business, Games
Publications: MICRO, Byte, 80 US, Compute

Omni Unlimited
105 S. Los Robles
Pasadena, California 91101
Contact: Don K. Fenstermaker
213/795-6664
Hardware: CBM, PET, Billings
Software: Educational, Personal, Business, Games, Legal, Medical, Dental, MFG
Publications: MICRO

Computer Systems Design Group
3632 Governor Drive
San Diego, California 92122
Contact: Mary Elizabeth Kroening
415/856-1954
Software: CAI/Adventure Authoring system for Apple II named "GROW"

Computer Business Systems
22963 La Cadena
Laguna Hills, California 92653
Contact: Lou
951-4640
Hardware: OSI, Data General
Software: For same

Desert Data & Electronics
212 Balsam (PO Box 1396)
Ridgecrest, California 93555
Contact: Larry Jenkins
714/375-4818
Hardware: Atari, CBM, OSI, PET, TI, Northstar, Dynabyte
Software: Educational, Personal, Business, Games
Publications: MICRO, Byte, Creative Computing, Kilobaud

FORTH Interest Group
P.O. Box 1105
San Carlos, California 94070
415/962-8653
(Mail Order)
Products/Services: Source Listing for 6502

Sunset Electronics
2254 Taraval St.
San Francisco, California 94116
Contact: John Warburton
415/665-8330
Hardware: OSI
Software: Educational, Personal, Business, Games
Publications: All computer

Computerland
22634 Foothill Blvd.
Hayward, California 94541
538-8080
Hardware: Apple, Atari, CBM, PET
Software: Educational, Personal, Business, Games
Publications: MICRO, Byte, 80 US, 80 Microcomputing, Kilobaud, Creative Interface

Computer Scene
753 So. State St.
Ukiah, California 95482
Contact: A.D. Rorabaugh
707/462-1578
Hardware: Atari, CBM, PET
Software: Educational, Personal, Business, Games

Oregon

Computerland of Portland
12020 S.W. Main
Tigard, Oregon 97223
Contact: Cliff Bihl
503/620-6170
Hardware: Apple, Atari, CBM, PET, Cromemco, Vector
Software: Educational, Personal, Business, Games
Publications: MICRO, Byte, etc.

Washington

The Electronics Shop
131 North Decatur
Olympia, Washington 98502
Contact: Tim
206/357-6304
Hardware: Atari, CBM, PET, S-100
Software: Educational, Personal, Business, Games
Publications: MICRO, Byte, Interface Age, Kilobaud, 80 Microcomputing, S-100 Microsystems, Creative Computing

FOREIGN

Canada

(Ontario)

Computer Innovations Head Office
1719 St. Laurent Blvd.
Ottawa, Ontario, Canada K1G 3V4
Contact: Kathryn Miller, Purchasing or Bruce McLaws, Sales
Hardware: Apple, CBM, PET, Cromemco
Software: Educational, Personal, Business, Games
Publications: Byte, Compute, Kilobaud, Nibble, etc.

House of Computers Inc.
368 Eglinton Avenue West
Toronto, Ontario, Canada M5N 1A2
Contact: Anita Tchoryk
Hardware: Apple, CBM, KIM, PET
Software: Educational, Personal, Business, Games
Publications: MICRO, Byte, Interface Age, Microcomputing, 80 Magazine, Nibble, Compute, Creative Computing

The Computer Centre
259 Davis Street
Sarnia, Ontario N7T 1B4
519/337-6332
Publications: MICRO and others

(British Columbia)

Conti Electronics Ltd.
7204 Main Street
Vancouver, British Columbia, Canada V5X 3J4
Contact: Gary W. Skett
604/324-0505
Hardware: PET
Software: Educational (limited), Personal, Business, Games, Industrial Monitoring and Testing Hardware
Publications: MICRO, Compute, Creative Computing

Australia

Electronic Concepts Pty. Ltd.
55 Clarence Street
Sydney, N.S.W. Australia 2000
Contact: D.T. or L. Hoess
Hardware: Apple, Onyx
Software: Educational, Personal, Business, Games, Scientific
Publications: MICRO, Byte, Interface, Personal Computing, Kilobaud

Belgium

MICRO 2000
32 Galerie D'Ixelles
Brussels, Belgium 1050
Contact: J. Marsily
512 4305
Hardware: Apple, TRS-80
Software: Educational, Personal, Business, Games
Publications: MICRO, Nibble, 80-US, Call APPLE, Compute

Compulec
Naamsesteenweg 382/W5
3030 Leuven (Heverlee)
Belgium 3030
Contact: Y. Rummens
16-22.34.88
Hardware: AIM, Apple, Sharp-ITT 2020
Software: Educational, Personal, Business

Netherlands

Computercollectief
Amstel 312a
1017 AP Amsterdam, Netherlands
Contact: Frans de Vreeze
020-223573
Software: Educational, Personal, Business, Games (for Apple, Sorcerer and TRS-80)
Publications: At least 30 magazines, 300 books

France

Alpha Systemes
51 Rue Thiers
Grenoble, France 38000
Contact: P. Binet
76.87.98.27
Hardware: Apple, OSI, Sharp, Hewlett-Packard
Software: Educational, Personal, Business, Games
Publications: Many

SIVEA
31 BD des Batignolles
Paris, France 75008
Contact: J. Cleenewerck
522 70 66
Hardware: Apple, TRS-80
Software: Educational, Personal, Business, Games
Publications: MICRO, Nibble, Call APPLE, Byte, Creative Computing, 80-US, etc.

MICRO PET Vet

By Loren Wright

Numbering of BASIC Versions

There seems to be a bit of confusion on the numberings of BASIC versions. One popular numbering system for the BASICs is the one I have been following: 2.0, 3.0, and 4.0. However, the one used by Commodore is a little different. The original BASIC (the "old" ROM's) is called 1.0. The "upgrade" version, produced until recently, is 2.0, and now we have 4.0. I will use the latter convention from now on, and will edit manuscripts published in MICRO accordingly.

All 80-column machines have 4.0 ROMs, as do recent production 40-column machines. These new 40-column machines are called 4016 and 4032, but the actual circuit board is still the same. Some of the enclosures have plastic tops, apparently enlarged to accommodate a disk drive. The disk drive idea doesn't seem to have caught on, but I expect we'll be seeing more and more plastic tops. Also, boards continue to be delivered with holes drilled in the traces of 4016 PC boards where the second row of RAM chips would go. This is to prevent users or unscrupulous dealers from making a cheap and easy memory upgrade.

There is an upgrade kit to go from 2.0 to 4.0 BASIC, and you could even go from 24-pin 1.0 ROMs to 4.0 if you had another socket to install the eighth chip. As far as I know, there is no 28-pin (6540) upgrade kit.

There are several good reasons to upgrade—faster garbage collection and more powerful disk commands, for instance. All but the most serious programmers will stay with what they have. After all, the old ROMs weren't so bad were they?

Toward Universal PET Programs

MICRO will continue to publish articles for all three BASIC ROM sets, and for both disk operating systems, but as I mentioned in a previous column, articles that apply to all three are much preferred.

I call your attention to "PRINT USING for the PET" by David Malmberg in this issue. Not only has he reworked an excellent Apple program for the PET, but he has also accommodated all three PET ROM sets. This involved knowing the right page-zero locations and system calls for each ROM set. These are available from the memory maps and entry point lists published by Jim Butterfield in *Compute* and *The Transactor*. Malmberg also uses the contents of 50003 to identify which BASIC is being used: 0—1.0; 1—2.0; 160—4.0. Because the numbers involved are easy to remember, this is fast becoming a standard technique. Some other frequently-used locations that vary from BASIC to BASIC are given in table 1.

Page zero locations tend to be the same in BASIC 2.0 and 4.0, but in 1.0 they are completely different. PET system calls have different addresses, but generally they work similarly in each BASIC. Malmberg's BASIC program will run on any PET or CBM machine without modification.

80-Column Functions

The 80-column function table that appeared in last month's column contained some errors. The corrected version, with a couple of additions, is shown in table 2.

The window feature on the 80-column machines can be very powerful. It confines user input (and the computer's attention) to a restricted area of the screen. The SET TOP and SET BOTTOM commands fix the upper left and lower right corners of the window. The window may also be defined by POKEing four values into memory for the four edges:

	Address	Range
TOP	224	0 to 24
BOTTOM	225	TOP to 24
LEFT	226	0 to 79
RIGHT	213	LEFT to 79

The window may be cleared by printing or striking on the keyboard two successive HOMEs.

Table 1

	1.0	2.0	4.0
1) End of memory pointer	134,135	52,53	52,53
2) # characters in keyboard buffer	525	158	158
3) Disable STOP key POKE	537,136	144,49	144,88
4) Enable STOP key POKE	537,133	144,46	144,85

Table 2

Function	ASCII	Reverse Field Character	Keyboard Combination
BELL	7	g	
DELETE LINE	21	u	ESC, RVS, K
ERASE to BEGINNING of line	150	V	LS,←, 3
ERASE to END of line	22	v	←, Q, 4
GRAPHICS screen	142	N	LS, RS
INSERT line	149	U	SH, ESC, RVS, K
SCROLL DOWN	153	Y	LS, ESC, K
SCROLL UP	25	y	
SET BOTTOM	143	O	SH, Z, A, L
SET TOP	15	o	Z, A, L
SET TAB/CLEAR TAB	137	I	SH, TAB
TAB	9	i	TAB
TEXT screen	14	n	

SH = either shift
LS = left shift
RS = right shift
All digits are on the numeric keypad, not the main keyboard.

Big Savings On Atari & PET!

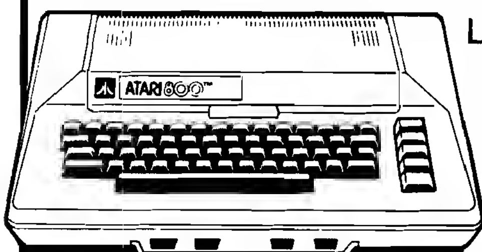
No Risk - No Deposit On Phone Orders - Shipped Same Day You Call° - C.O.D. or Credit Card

* On all in stock units

Please Call Between 11AM & 6PM (Eastern Standard Time)

(800) 233-8950

ATARI® 800™ PERSONAL COMPUTER

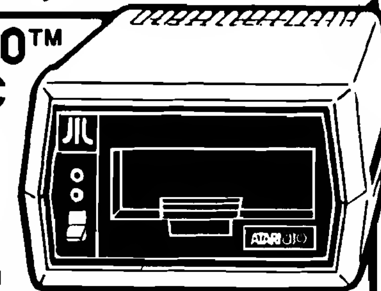


List $1080

$759

ATARI®

ATARI® 810™ DISC DRIVE



List $599.95

New Low Price only $489.00

Maxell Disks	10 for $34
Sycom Disks	10 for 29
Atari Disks	5 for 22

400 8K	$419
400 16K	449
410 Recorder	62
815 Disk	1199
822 Printer	359
825 Printer	779
830 Modem	159
850 Interface Module	179
CX852 8K RAM	94
CX853 16K RAM	149
CX70 Light Pen	64
CX30 Paddle	18
CX40 Joystick	18
CX86 Printer Cable	42
CO16345 822 Thermal Printer Paper	5
CAO16087 825 80-col. Printer Ribbon (3/box)	17
CX4119 Conversational French	45
CX4118 Conversational German	45
CX4120 Conversational Spanish	45
CX4125 Conversational Italian	45
CXL4009 Chess	30
CXL4011 Star Raiders™	45
CXL4004 Basketball	30
CXL4006 Super Breakout™	30
CXL4010 3-D Tic-Tac-Toe	30
CXL4005 Video Easel™	30
CXL4007 Music Composer	45
CXL4015 TeleLink™	20
CXL4002 BASIC Computing Language	46
CXL4001 Education System Master	21
CXL4003 Assembler Editor	45

CX4115 Mortgage & Loan Analysis	$13
CX4101 An Invitation to Programming 1	17
CX4106 An Invitation to Programming 2	20*
CX4117 An Invitation to Programming 3	20*
CX4107 Biorhythm	13
CX4103 Statistics I	17*
CX4121 Energy Czar	13
CX4108 Hangman	13
CX4102 Kingdom	13
CX4112 States & Capitals	13
CX4114 European Countries & Captials	13
CX4105 Blackjack	13
CX4111 Space Invaders	18

* These are scheduled for release in the first quarter of 1981.

Talk & Teach Courseware;
CX6001 thru CX6017 23

Atari Specialists . . . We carry it all!

Combination Special!

825 Printer & 850 Interface

825 Lists for $1000
Regular Mail Order Price: $779
850 Lists for $220.
Regular Mail Order Price: $179

Save $20.00

Buy Both For Only $938

CX8104 Atari 810 Master Diskette II

New DOS 2 Operating System Master

only $21.00

Microtek RAM 16K or 32K

- Full 1 year warranty
- Compatible with 400 or 800
- Assembled and Tested

16K	$ 99
32K	189

Atari March Releases:

CX8106 Bond Analysis	$20
CX8107 Stock Analysis	20
CX8108 Stock Charting	20
CX4104 Mailing List	17
CX4110 Touch Typing	20
CX8102 Calculator	24
CX4109 Graph It	17
CX4120 Conversational Spanish	45

Software

WordPro 3 (40 col.)	$186
WordPro 4 (80 col.)	279
WordPro 4 Plus (80 col.)	339
Visicalc - Apple	$122
Atari	163
PET	163

commodore

Commodore Computers:

4032 N	$1090
4032 B	1090
8032	1499

Commodore Peripherals:

CBM 2022 Printer	675
CBM 4040 Dual Drive Floppy Disk	1090
CBM 8050 Dual Drive Floppy Disk	1420
CBM C2N Cassette Drive	87
Tally 8024 Printer	1679

Commodore Software

EBS Accounts Receivable Inventory System	$695
Dr. Daley Mailing List	129
Dr. Daley Inventory	89
OZZ Information System	329
BPI General Ledger	329
Tax Package	399
Dow Jones Portfolio Management	129
Pascal	239

Cables:

PET to IEEE Cable	37
IEEE to IEEE Cable	46

Printers

NEC 5530	$2495
Diablo 630	2195
Trendcom 100	299
Trendcom 200	489
Paper Tiger 445G	769
Paper Tiger 460G	1219
Epson MX-80	539

To Order:

Phone orders invited (800 number is for order desk only). Or send check or money order. Equipment Shipped UPS collect. Pennsylvania residents add 6% sales tax. Add 3% for Visa or MC. Equipment is subject to price change and availability without notice.

Computer Mail Order
501 E. Third St.
Williamsport, PA 17701
(717) 323-7921

Add a Light Pen to your Micro

This article includes the hardware details necessary to install a light pen on any 6502 system. Software is included for an OSI implementation.

Peter Alan Koski
27 Dogwood Drive
Allendale, New Jersey 07401

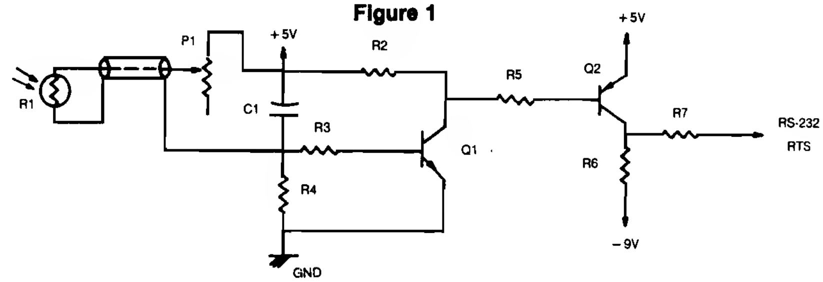
Many computer installations today offer alternate forms of user I/O other than the standard CRT/keyboard combination. Among these is the light pen. In using a light pen, the user, if choosing from a menu for example, simply points the pen at what he desires. When locating a point on a grid, the user simply locates the point using the pen, rather than inputting coordinates through the keyboard.

Although the user may find this simplification of input fabulous, for the programmer there exists a lot of overhead. The programmer must keep track of where the information is located on the screen as the program progresses, and how the information changes during program execution.

Principles of Operation

In theory, the operation of a light pen is extremely straightforward. When a request is made to locate the pen, a distinguishable token is swept across the display until the pen recognizes its presence. At that time, if we know where the token is, we also know where the pen is. Simple as this may seem, the hardware and software doesn't always follow suit on simplicity. On graphics systems where there is often a stand-alone microprocessor to control the terminal functions, the "token" is the raster sweep. This is

Figure 1



Bill of Materials

R1 — photo-resistor (see text)
R2 — 1.8K
R3 — 18K
R4 — 120K
R5, R6 — 10K
R7 — 470
Q1 — 2N5300 (RS 276-2009)
Q2 — 2N5226 (RS 276-2032)
C1 — .005 uF
P1 — 100K PC-type potentiometer

Table 1

0 D000	1 D008	2 D010	3 D018	4 D020	5 D028	6 D030	7 D038
8 D100	9 D108	10 D110	11 D118	12 D120	13 D128	14 D130	15 D138
16 D200	17 D208	18 D210	19 D218	20 D220	21 D228	22 D230	23 D238
24 D300	25 D308	26 D310	27 D318	28 D320	29 D328	30 D330	31 D338
32 D400	33 D408	34 D410	35 D418	36 D420	37 D428	38 D430	39 D438
40 D500	41 D508	42 D510	43 D518	44 D520	45 D528	46 D530	47 D538
48 D600	49 D608	50 D610	51 D618	52 D620	53 D628	54 D630	55 D638
56 D700	57 D708	58 D710	59 D718	60 D720	61 D728	62 D730	63 D738

probably the most sophisticated and elegant approach since the resolution is extremely high and the scan is invisible to the user.

I have taken a much more simplified approach. Rather than search the entire screen for the pen's location, I request verification at given screen locations. With this approach, the token must be displayed on the screen in order for the pen to see it. Since the standard scan rate for a monitor is 1/60 second, we have to display the token and then wait the required 1/60 second to guarantee that the token reaches the display. Obviously if we were to poll 2048 display locations, the time required wouldn't make this practical.

Hardware

Figure 1 and the accompanying "bill of materials" presents the design for the light pen circuit I am using. The sensor used is a small Calectro photo-resistor mounted in a magic marker casing. The choice of the photo-resistor over a photo-transistor was based on spectrum sensitivity. Photo-transistors that I found were not responsive to a phosphorus source. The photo-resistor was, so the choice was obvious.

The light/dark conditions are reflected via an RS-232 line which is toggled between +5 and −9 volts. RS-232 was chosen as the communications link since I have an RS-232 port on my machine (as do most). By using the light pen to drive the RTS line on the port, I can monitor the status of the pen by monitoring the status word of the ACIA. The pen's condition will be found at the RTS bit.

The circuit goes together nicely on a small piece of perf-board. The interconnecting line used is a piece of miniature shielded micro-phone cable. There shouldn't be any problem assembling the circuit and all that remains is to adjust the light pen to match the CRT used.

A BASIC routine can be used to initially align the pen. While printing the contents of the status port, adjust the monitor's brightness/contrast and P1 of the light pen circuit. A point should be found so that touching the pen to an illuminated position will cause the RTS bit to be set low ("0"), a dark position should set the RTS bit high ("1"). It should be possible to find a position which is comparable to normal viewing intensity.

```
0800              ;****************************
0800              ;*                          *
0800              ;* LIGHT PEN QUADRANT QUERY *
0800              ;*         ROUTINE          *
0800              ;*                          *
0800              ;*      PETER A. KOSKI      *
0800              ;*                          *
0800              ;****************************
0800              ;
0800              ;
3280                      ORG $3280
3280              PENWRD  EQU $FC00
3280 206C33               JSR GETBLK
3283              ;
3283              ;GET USR ARGUMENT (QUADRANT NUMBER) -- INSU
RE
3283              ;THAT THE VALUE IS ONLY 0 - 63
3283              ;
3283 A5B2                 LDA $B2
3285 293F                 AND #%00111111
3287 A8                   TAY                  ;GET ADDR LOOKUP
 OFFSET
3288              ;
3288              ;LOAD BASE ADDRESS INTO STORE/BLANK/SCAN/RE
STR
3288              ;ROUTINES
3288              ;
3288 B9BA33               LDA LOADDR,Y
328B 8DBB32               STA STORE+1
328E 8DC332               STA BLANK+1
3291 8D0233               STA SCRN1+1
3294 8D3D33               STA RESTRE+1
3297              ;
3297 B97A33               LDA HIADDR,Y
329A 8DBC32               STA STORE+2
329D 8DC432               STA BLANK+2
32A0 8D0333               STA SCRN1+2
32A3 8D3E33               STA RESTRE+2
32A6              ;
32A6              ;INITIALIZE ADDRESSES FOR SAVE BUFFER
32A6              ;BUFFER STARTS AT $337E
32A6              ;
32A6 A97A                 LDA #$7A
32A8 8DBE32               STA SCRN+1
32AB 8D3A33               STA SCRN2+1
32AE A933                 LDA #$33
32B0 8DBF32               STA SCRN+2
32B3 8D3B33               STA SCRN2+2
32B6              ;
32B6              ;SAVE BLOCK DATA/BLANK (DARKEN) SELECTED
32B6              ;QUADRANT
32B6              ;
32B6 A000                 LDY #00
32B8 A200         SAVE    LDX #00
32BA BDFFFF       STORE   LDA $FFFF,X
32BD 8DFFFF       SCRN    STA $FFFF            ;SAVE CHAR IN BU
FFER
32C0 A920                 LDA #$20
32C2 9DFFFF       BLANK   STA $FFFF,X
32C5 EEBE32               INC SCRN+1
32C8 D003                 BNE NOCRY1
32CA EEBF32               INC SCRN+2
32CD E8           NOCRY1  INX
32CE E008                 CPX #08              ;8 CHAR/LINE/QUA
D
32D0 D0E8                 BNE STORE
32D2 C8                   INY
32D3 C004                 CPY #04              ;4 LINE/QUAD
32D5 F01A                 BEQ OUT1
32D7 ADBB32               LDA STORE+1
32DA 18                   CLC
32DB 6940                 ADC #$40
32DD 8DBB32               STA STORE+1
32E0 8DC332               STA BLANK+1
32E3 ADBC32               LDA STORE+2
32E6 6900                 ADC #00
32E8 8DBC32               STA STORE+2
32EB 8DC432               STA BLANK+2          ;GET NEXT LINE A
DDR
32EE 4CB832               JMP SAVE
32F1              ;
```

```
32F1                ;SEE IF LIGHT PEN WENT DARK
32F1                ;
32F1 206F33         OUT1    JSR TVDLA           ;1/60 S. SCAN DE
LAY
32F4                ;
32F4 AD00FC                 LDA PENWRD
32F7 2908                   AND #%00001000
32F9 F037                   BEQ NOTFND          ;DARK PEN SETS R
TS
32FB                ;
32FB                ;PEN IS DARK/ENABLE QUADRANT AND SEE IF
32FB                ;PEN SEES CHANGE
32FB                ;
32FB A000                   LDY #00
32FD A200           ENABLE  LDX #00
32FF A9A1                   LDA #$A1            ;WHITE SQUARE CH
AR.
3301 9DFFFF         SCRN1   STA $FFFF,X
3304 E8                     INX
3305 E408                   CPX $08
3307 D0F8                   BNE SCRN1
3309 C8                     INY
330A C004                   CPY #04
330C F014                   BEQ OUT2            ;ENTIRE QUAD ENA
BLED
330E AD0233                 LDA SCRN1+1
3311 18                     CLC
3312 6940                   ADC #$40
3314 8D0233                 STA SCRN1+1
3317 AD0333                 LDA SCRN1+2
331A 6900                   ADC #00
331C 8D0333                 STA SCRN1+2
331F 4CFD32                 JMP ENABLE          ;ADDR OF NEXT LI
NE
3322                ;
3322                ;CHECK TO SEE IF PEN SEES ENABLED QUADRANT
3322                ;
3322 206F33         OUT2    JSR TVDLA           ;1/60 S. SCAN DE
LAY
3325                ;
3325 AD00FC                 LDA PENWRD
3328 2908                   AND #%00001000
332A D006                   BNE NOTFND          ;HI PEN DISABLES
 RTS
332C                ;
332C                ;RETURN CODE FOR PEN:
332C                ;1 TO BASIC = PEN WAS IN QUADRANT
332C                ;0 TO BASIC = PEN WAS NOT IN QUAD
332C                ;
332C A901                   LDA #01
332E 48                     PHA                 ;SAVE RETURN COD
E ON STACK
332F 4C3533                 JMP REPLCE
3332                ;
3332 A900           NOTFND  LDA #00
3334 48                     PHA                 ;SAVE RETURN COD
E ON STACK
3335                ;
3335                ;RESTORE ORIGINAL DATA FOUND AT QUADRANT
3335                ;
3335 A000           REPLCE  LDY #00
3337 A200           RPLCE   LDX #00
3339 ADFFFF         SCRN2   LDA $FFFF
333C 9DFFFF         RESTRE  STA $FFFF,X
333F EE3A33                 INC SCRN2+1
3342 D003                   BNE NOCRY2
3344 EE3B33                 INC SCRN2+2
3347 E8             NOCRY2  INX
3348 E008                   CPX #08
334A D0ED                   BNE SCRN2
334C C8                     INY
334D C004                   CPY #04
334F F014                   BEQ OUT3
3351 AD3D33                 LDA RESTRE+1
3354 18                     CLC
3355 6940                   ADC #$40
3357 8D3D33                 STA RESTRE+1
335A AD3E33                 LDA RESTRE+2
335D 6900                   ADC #00
```

Software

The two routines presented here are essentially identical except for the resulting resolution. Both are called via the BASIC USR function. The longer of the two routines accepts argument values from 0-63, the number corresponding to the screen quadrant to be queried. Table 1 shows the quadrant numbering scheme. The address associated with each quadrant is the address of the upper left memory location in the quadrant. Quadrants run eight locations horizontally and four locations vertically, or 32 locations total. Thus, touching the pen to any of these locations will score a hit. A hit is returned to BASIC as a 1 from the USR function, a miss is returned as a 0. This routine is thus most useful when resolution is not critical, such as for menu selection.

The single-cell query routine polls individual memory locations and thus provides 64 × 32 resolution. The argument of the USR function should be the requested memory address, less 32768. (BASIC only allows signed *15*-bit arguments.) The return value is the same as the previous: 1 if hit, 0 if miss.

Both routines use the same idea in polling the requested position. The information at the quadrant or single cell is first saved and replaced by OSI graphics character $20 (blank). If the pen is presently looking at a dark location, we *might* have its position. If not, we replace the data and return a miss. Should the pen be dark, we replace the $20 with $A1 (full illumination character). At this point, if the pen sees a transition to light, we are at the correct position and return a hit after restoring the data. Had the transition not been seen by the pen, we obviously were not at the right location, and would return a miss.

Programming with a Light Pen

When using the light pen, screen locations become very critical, thus careful formatting should be used through the memory map supplied by OSI. Remember that when using standard input and print statements, the screen has a tendency to scroll. Fortunately, this can be avoided by disabling the line feed. POKE 9644,42 will disable the scroll, POKE 9644,98 will re-enable the scroll routine.

(continued on page 63)

(Add a Light Pen...)

```
335F 8D3E33   146          STA RESTRE+2
3362 4C3733   147          JMP RPLCE
3365          148   ;
3365          149   ;RETURN TO BASIC CALLING ROUTINE
3365          150   ;
3365 68       151   OUT3   PLA                    ;POP RETURN CODE

3366 A8       152          TAY
3367 A900     153          LDA #00
3369 6C0800   154          JMP (08)               RTS
336C          155   ;
336C 6C0600   156   GETBLK JMP (06)
336F          157   ;
336F          158   ;TV SCAN DELAY
336F          159   ;
336F A040     160   TVDLA  LDY #$40
3371 A2FF     161   LOOP1  LDX #$FF
3373 CA       162   LOOP2  DEX
3374 D0FD     163          BNE LOOP2
3376 88       164          DEY
3377 D0F8     165          BNE LOOP1
3379 60       166          RTS
337A          167   ;
337A          168   ;QUADRANT CHARACTER HOLD BUFFER
337A          169   ;
337A          170   BUFFER EQU *+32
337A          171   ;
337A          172   ;ADDRESS LOOK UP TABLE FOR 64 GIVEN QUADRAN
TS
337A          173   ;
337A D0D0D0   174   HIADDR HEX D0D0D0D0D0D0D0D0
337D D0D0D0
3380 D0D0
3382 D1D1D1   175          HEX D1D1D1D1D1D1D1D1
3385 D1D1D1
3388 D1D1
338A D2D2D2   176          HEX D2D2D2D2D2D2D2D2
338D D2D2D2
3390 D2D2
3392 D3D3D3   177          HEX D3D3D3D3D3D3D3D3
3395 D3D3D3
3398 D3D3
339A D4D4D4   178          HEX D4D4D4D4D4D4D4D4
339D D4D4D4
33A0 D4D4
33A2 D5D5D5   179          HEX D5D5D5D5D5D5D5D5
33A5 D5D5D5
33A8 D5D5
33AA D6D6D6   180          HEX D6D6D6D6D6D6D6D6
33AD D6D6D6
33B0 D6D6
33B2 D7D7D7   181          HEX D7D7D7D7D7D7D7D7
33B5 D7D7D7
33B8 D7D7
33BA          182   ;
33BA 000810   183   LOADDR HEX 0008101820283038
33BD 182028
33C0 3038
33C2 000810   184          HEX 0008101820283038
33C5 182028
33C8 3038
33CA 000810   185          HEX 0008101820283038
33CD 182028
33D0 3038
33D2 000810   186          HEX 0008101820283038
33D5 182028
33D8 3038
33DA 000810   187          HEX 0008101820283038
33DD 182028
33E0 3038
33E2 000810   188          HEX 0008101820283038
33E5 182028
33E8 3038
33EA 000810   189          HEX 0008101820283038
33ED 182028
33F0 3038
33F2 000810   190          HEX 0008101820283038
33F5 182028
33F8 3038
```

```
0800           ;***************************
0800           ;*                         *
0800           ;* SINGLE VIDEO CELL QUERY *
0800           ;*        ROUTINE          *
0800           ;*                         *
0800           ;*     PETER A KOSKI       *
0800           ;*                         *
0800           ;***************************
0800           ;
3280                  ORG $3280
3280                  OBJ $800
3280           PENWRD EQU $FC00
3280 20C432           JSR GETBLK
3283           ;
3283           ; TURN USR ARGUMENT INTO 6502 ADDRESS
3283           ;
3283 A5B2             LDA $B2
3285 48               PHA
3286 A5B1             LDA $B1
3288 0980             ORA #%10000000
328A 85B2             STA $B2
328C 68               PLA
328D 85B1             STA $B1
328F           ;
328F A200             LDX #00
3291           ;
3291           ;SAVE CHARACTER AT CELL AND DARKEN
3291           ;SELECTED CELL
3291           ;
3291 A1B1             LDA ($B1,X)
3293 48               PHA                 ;SAVE CHAR ON ST
ACK
3294 A920             LDA #$20
3296 81B1             STA ($B1,X)
3298           ;
3298           ;SEE IF LIGHT PEN WENT DARK
3298           ;
3298 20C732           JSR TVDLA           ,1/60 S. SCAN DE
LAY
329B AD00FC           LDA PENWRD
329E 2908             AND #%00001000
32A0 F018             BEQ NOTFND          ;DARK PEN SETS R
TS
32A2           ;
32A2           ;PEN IS DARK / ENABLE CELL AND CHECK IF
32A2           ;PEN SEES TRANSITION
32A2           ;
32A2 A9A1             LDA #$A1            ;WHITE SQUARE CH
AR.
32A4 81B1             STA ($B1,X)
32A6           ;
32A6           ;CHECK TO SEE IF PEN SAW ENABLED CELL
32A6           ;
32A6 20C732           JSR TVDLA           ;1/60 S. SCAN DE
LAY
32A9           ;
32A9 AD00FC           LDA PENWRD
32AC 2908             AND #%00001000
32AE D00A             BNE NOTFND          ;HI PEN DISABLES
 RTS
32B0           ;
32B0           ;RETURN CODE FOR PEN:
32B0           ;1==PEN WAS AT CELL
32B0           ;0==PEN WAS NOT AT CELL
32B0           ;
32B0 68               PLA                 ;GET CELL'S CHAR
ACTER
32B1 81B1             STA ($B1,X)
32B3 A900             LDA #00
32B5 A001             LDY #01
32B7 6C0800           JMP (08)
32BA           ;RTS -- BASIC
32BA           ;
32BA 68        NOTFND PLA                 ;GET CELL'S CHAR
ACTER
32BB 81B1             STA ($B1,X)
32BD A900             LDA #00
32BF A000             LDY #00
32C1 6C0800           JMP (08)
32C4           ;RTS -- BASIC
```

(Add a Light Pen...)

```
32C4               ;
32C4 6C0600        GETBLK JMP (06)
32C7               ;
32C7               ;DELAY TO ALLOW FOR STANDARD TV SCAN
32C7               ;
32C7 A000          TVDLA  LDY #00
32C9 A200          LOOP1  LDX #00
32CB E8            LOOP2  INX
32CC D0FD                 BNE LOOP2
32CE C8                   INY
32CF C040                 CPY #$40
32D1 D0F6                 BNE LOOP1
32D3 60                   RTS
```

```
 950 REM  ***************************
 952 REM  *                         *
 954 REM  * LIGHT PEN DEMONSTRATION *
 956 REM  *     BY PETER A KOSKI    *
 958 REM  *                         *
 960 REM  ***************************
1004 REM  CALL IN SCREEN CLEAR ROUTINE
1005 REM
1006 DISK!"CALL 3280=31,3"
1010 POKE 574,128: POKE 575.50
1020 X=USR(X)
1021 REM
1022 REM LOAD SELECTION DATA ONTO SCREEN
1023 REM
1030 PRINT "           == ELEMENTARY MATH LIGHT PEN DEMO =="

1040 PRINT : PRINT : PRINT
1050 PRINT "  0     1     2     3     4     5     6     7
 8";
1060 PRINT "     9"
1070 PRINT : PRINT : PRINT : PRINT
1080 PRINT "        +      -        *       /"
1090 PRINT : PRINT : PRINT : PRINT . PRINT
2000 REM
2010 REM    RUN THE PROGRAM USING SUBROUTINES
2020 REM
2021 REM       CALL IN SINGLE CELL QUERY ROUTINE
2022 REM
2025 DISK!"CALL 3280=31,2":RPT=0
2030 GOSUB 3000:A1=NUM
2040 GOSUB 4000:O=OP:RPT=RPT+1
2050 GOSUB 3000:A2=NUM
2060 ON O GOTO 2100,2200,2300,2400
2100 ANS=A1+A2
2110 GOTO 2500
2200 ANS=A1-A2
2210 GOTO 2500
2300 ANS=A1*A2
2310 GOTO 2500
2400 ANS=A1/A2
2500 REM
2510 REM      PRINT OUTPUT
2520 REM
2530 POKE 9644,42: REM   DISABLE SCROLL ON PRINT
2600 PRINT " THE RESULT IS ";ANS
2605 FOR PS=1 TO 200:PS= ABS (PS): NEXT PS
2620 PRINT "                                   "
2625 IF RPT=5 THEN 5000
2630 GOTO 2030
3000 REM
3010 REM    NUMERIC INPUT (SINGLE DIGITS)
3020 REM
3025 NUM=0:CNT=0
3030 FOR CELL=54402 TO 54458 STEP 6
```

```
3040 NUM=USR(CELL-32768)
3050 IF NUM=1 THEN NUM=CNT: RETURN
3060 CNT=CNT+1
3100 NEXT CELL
3110 CNT=0
3120 GOTO 3030
4000 REM
4010 REM   OPERATOR INPUT
4020 REM
4025 CNT=1
4030 FOR CELL=54730 TO 54760 STEP 10
4040 OP=USR(CELL-32768)
4045 IF OP=1 THEN OP=CNT: RETURN
4046 CNT=CNT+1
4050 NEXT CELL
4100 GOTO 4025
5000 REM
5010 REM   ANOTHER 5 ROUNDS ?
5020 REM
5021 REM   CALL IN SCREEN CLEAR
5022 REM
5030 DISK!"CALL 3280=31,3
5040 X=USR(X)
5045 REM
5046 REM CALL IN QUADRANT QUERY ROUTINE
5047 REM
5050 DISK!"CALL 3280=31,1
5060 POKE 9644,98: REM  RE-ENABLE SCROLL
5065 REM
5066 REM           DISPLAY QUESTION/INPUT SELECTION
5067 REM
5070 PRINT "   WOULD YOU LIKE ANOTHER GO AT IT ?"
5080 PRINT : PRINT : PRINT : PRINT
5090 PRINT "           YES                    NO"
5100 PRINT : PRINT
5110 IF USR(49)=1 THEN 950
5120 IF USR(52)=1 THEN 6000
5130 GOTO 5110
6000 REM
6010 REM   EXIT TO BASIC
6020 REM
6030 DISK!"CALL 3280=31,3
6040 X=USR(X)
6050 END
```

Other thoughts to keep in mind, especially if taking input from a graph, is that the pen can move only to a block adjacent to the one where it is presently. Thus, once the original position is known, the next move can only be one of, at most, eight positions. The accompanying demonstration program should help to explain. The two routines and a screen clear are on track 31 in this example, and are all called into $3280 for execution. The quadrant routine is on sector 1, the individual cell routine on sector 2, and the screen clear code on sector 3. The program is a simple arithmetic demo. Single-digit arguments and operators are input via the light pen and the result is printed to the screen without line feed. After five repetitions, the program asks the user if he would like another session. Again, the yes/no response is accepted through the light pen.

Conclusion

Although the routines presented were written for an OSI mini-floppy system, any 6502 system supporting memory mapped video should be able to employ them. The only changes to be made are the addresses of the display block and of the ACIA.

One last thought for disk owners—a menu or display block which is repeatedly used at various points of the program may be held resident on a disk track and then CALLed to $D000. As you'll discover, speed plays an important part in light pen I/O, since the time spent for input is nil. The processing and output must therefore be as streamlined as possible in developing an efficient system.

MICRO

Integer BASIC Internals

Here's a sorted list of Apple Integer BASIC memory locations and routines, with some examples of how to use them.

Glenn R. Sogge
P.O. Box 203
Evanston, Illinois 60204

Apple Computer Co. has released to its dealers a set of application and information notes that are quite informative. Included in the package is a listing of memory locations and routines used by Integer BASIC. The table with this article is a numerically sorted (by hex address) listing of this information. Also included are the corresponding decimal equivalents of the addresses and a little information about the routines. In general, routines without descriptions are the handlers for the functions named.

Hardly any information is given about how the routines are called or are used but with a little digging I'm sure you can figure out how to use at least a few of these in your own programs. At least, you now know where to begin looking. For example, the routine at $E51B ("HEX/DEC") converts the 16-bit number contained in the X (lo) and A (hi) registers to a decimal number from 0-65535 and prints it out.

The routine at $EE68 prints out the "*** RANGE ERR" message and the routine at $E36B prints out the "*** MEM FULL ERR" message. A couple of useful tricks are also mentioned in the Apple material.

1. To find the absolute address of a given line, place the line number (in hex) into locations $CE-$CF (lo and hi). Then jump to address $E56D (*E56DG); the absolute address will be returned in locations $E4-$E5 (lo and hi).

2. To execute a GOTO from the monitor, put the absolute address of the line (found by the above procedure) into $C6-$C7 and put a negative byte into the mode location (eg., $80 into $D9—a negative value indicates run mode, a positive one indicates immediate mode). Then jump to $E867 (*E867G) and you will be back in BASIC running at that line.

This item was picked up from a bulletin board here in Chicago and is from Mark Pump.

"If you've ever accidentally pressed RESET while an Integer BASIC program was running, this is for you. In the monitor, enter:

*E3E3G

and the statement number which was last executed is displayed. Press RESET again and re-enter DOS with *3D0G. This method can also be used to find the statement number of an outstanding Integer BASIC input statement. When the input prompt occurs, press RESET and *E3E3G to find the statement number. Exiting the program with control-C will not show the statement number if an input statement was active."

You should also notice that there seems to be some discrepancy between the list of page zero locations used given in the list and the chart on page 75 of the new Apple II Reference Manual (the white book). According to the chart, locations $E0-$FF are not used by Integer BASIC but the detailed list shows this to be incorrect. Some of those locations are indeed used by BASIC! Also, some locations are used for a couple of things, depending on the routine in command, so the values might not always be what you would expect.

STD REPORT FOR REC#:0

HEX	DEC	NAME	DESCR
004A	74	LOMEML	LOW MEMORY LO
004B	75	LOMEMH	LOW MEMORY HI
004C	76	HIMEML	HIGH MEMORY LO
004D	77	HIMEMH	HIGH MEMORY HI
004E	78	RNDL	RANDOM # LO
004F	79	RNDH	RANDOM # HI
0050-006F	80-111	NOUNSTKL	NOUN STACK LO
0058-0077	88-119	SYNSTKH	SYNTAX STACK HI LOCS
0078-0097	120-151	NOUNSTKH	NOUN STACK HI
0080-009F	128-159	SYNSTKL	SYNTAX STACK LO LOCS
00A0-00BF	160-191	NOUNSTKC	NOUN STACK COUNTER
00A8-00C7	168-199	TXTNDXSTK	TEXT INDEX STACK
00C8	200	TXTNDX	TEXT INDEX VALUE
00C8	200	OUTVAL	OUTPUT VAL TEMP
00C9	201	YTEMP	TEMP FOR Y-REG
00C9	201	LEADBL	LEADING BLANKS INDEX
00CA	202	PPL	PGM PTR LO
00CB	203	PPH	PGM PTR HI
00CC	204	PVL	CURR VAR PNTR LO
00CD	205	PVH	CURR VAR PNTR HI
00CE	206	ACL	GEN'L ACC LO
00CE	206	VALGETL	PRIMARY EVAL TEMP LO
00CE-00CF	206-207	VAL	16-BIT TEMP FOR MATH
00CF	207	VALGETH	PRIMARY EVAL TEMP HI
00CF	207	ACH	GEN'L ACC HI
00D0	208	SRCHL	PTR FOR SEARCH VAR TBL LO
00D1	209	SRCHH	PTR FOR SEARCH VAR TBL HI
00D1-00F0	209-240	TOKNDXSTK	TOKEN INDEX STACK
00D2	210	SRCH2L	VAR TAB SEARCH PTR2 LO
00D3	211	SRCH2H	VAR TAB SEARCH PTR2 HI
00D4	212	IFSKIP	IF ? THEN FAIL FLAG
00D5	213	CRFLAG	CARR RTN FLAG
00D6	214	VERBNOW	CURR VERB IN USE
00D7	215	PRINOW	PRINT IT NOW FLAG
00D8	216	XSAVE	TEMP FOR X-REG
00D9	217	RUNMODE	RUN MODE FLAG
00DA	218	AUXL	AUX CNTR LO
00DB	219	AUXH	AUX CNTR HI
00DC	220	PRL	CURR LN VAL LO
00DD	221	PRH	CURR LN VAL HI
00DE	222	PNL	CURR NOUN PTR LO
00DF	223	PNH	CURR NOUN PTR HI

(continued)

00E0	224	PXL	CURR VERB PTR LO
00E1	225	PXH	CURR VERB PTR HI
00E2	226	P1L	AUX PTR1 LO
00E2	226	DELL	DELETE LN PTR LO
00E3	227	DELH	DELETE LN PTR HI
00E3	227	P1H	AUX PTR1 HI
00E4	228	FLAG	GEN'L FLAG BYTE
00E4	228	P2L	AUX PTR2 LO
00E4	228	LNAL	LINE # ADDR LO
00E5	229	LNAH	LINE # ADDR HI
00E5	229	P2H	AUX PTR2 HI
00E6	230	NXTL	NEXT PTR LO
00E6	230	P3L	AUX PTR3 LO
00E7	231	NEXH	NEXT PTR HI
00E7	231	P3H	AUX PTR3 HI
00F1	241	TOKNDX	TOKEN INDEX VAL
00F2	242	CONL	CONTINUE PTR LO
00F3	243	CONH	CONTINUE PTR HI
00F4	244	AUTOINCL	AUTO INC VAL LO
00F5	245	AUTOINCH	AUTO INC VAL HI
00F6	246	AUTOLNL	CURR AUTO LINE # LO
00F7	247	AUTOLNH	CURR AUTO LINE # HI
00F8	248	AUTOMODE	AUTO FLAG
00F9	249	COUNT	GEN'L CNTR BYTE
00F9	249	CHAR	CURR CHAR
00FA	250	LEADZR	LEADING ZEROS INDEX
00FB	251	FORNDX	FOR/NEXT LOOP INDEX
00FC	252	GOSUBNDX	GOSUB INDEX
00FD	253	SYNSTKNDX	SYNTAX STACK INDEX VAL
00FE	254	SYNPAGL	SYNTAX PAGE PTR LO
00FF	255	SYNPAGH	SYNTAX PAGE PTR HI
0200-02FF	512-767	INBUFF	INPUT BUFFER
E000	-8192	CNTLB	COLD ENTRY
E003	-8189	CNTLC	WARM ENTRY
E006	-8186	SETPRMPT	SET UP > PROMPT
E02A	-8150	NXTBYTE	GET NEXT BYTE 16BIT PNTR
E04B	-8117	LIST	LIST ALL
E050	-8099	LISTXY	LIST A RANGE
E06D	-8083	UNPACK	TOKENED CODE TO MNEMONICS
E130	-7888	DIMSTR	DIMENSION A STRING
E171	-7823	INPUTSTR	INPUT A STRING
E222	-7646	MULT	MULTIPLY
E27A	-7558	MOD	
E28A	-7542	SCRN	RETURN SCRN COLOR
E2B3	-7501	MAINLINE	MAIN COMPILE/EXEC CODE
E36B	-7317	MEMFUL	PRINTS '*** MEM FULL ERR'
E36F	-7313	DELETE	DELETE LINES X-Y
E3C0	-7232	ERRORMESS*	INPUT ERROR MSG
E3CE	-7218	GETCMD	GET KEYBOARD CMD
E3E0	-7200	ERRORMESS	PRNT ERR MSG GOTO MAINLINE
E51B	-6885	HEX/DEC	PRINT VAL(X=LO ACC=HI) 0-65535
E56D	-6803	LINADR	FIND LINE #'S ADDRESS
E5AD	-6739	NEW	
E5B7	-6729	CLR	
E6EC	-6420	BRANCH	GET LO/HI THEN JSR
E6FF	-6401	GETVERB	NEXT VERB TO USE
E715	-6379	GET16BIT	GET A 16-BIT VAL
E736	-6346	NOT	
E74A	-6326	ABS	
E75C	-6308	SGN	
E782	-6270	SUBTRACTION	
E785	-6267	ADDITION	
E7A1	-6236	TAB	
E7C1	-6207	COMMA	
E7E2	-6174	AUTO	AUTO LINE #
E828	-6104	IF/THEN	IF ? THEN ROUTINE
E83C	-6061	GOSUB	
E858	-6056	GOTO	
E867	-6041	GOLINE	GOTO LINE ADDR IN $C6-C7
E875	-6027	GETNEXT	FETCH NEXT TEXT STATEMENT
E8A5	-5979	RETURN	
E8C3	-5949	STOPPED AT	PRINT 'STOPPED AT LINE #'
E8D6	-5930	NEXT	NEXT END LOOP
E93A	-5830	FOR	FOR INITIAL ENTRY
E950	-5808	TO/FOR	LOOP CNTR # TO # STEP #
EA10-EA87	-5616 -5497	VERBADRL	VERB DISPATCH TAB LO
EA88-EAFF	-5496 -5377	VERBADRH	VERB DISPATCH TBL HI
EB00-EB99	-5376 -5223	MESSTXT	ERROR MESS. TEXT
EBAA	-5206	INPUT	INPUT ROUTINE
EC00-EDFF	-5120 -4609	SYNTABL	SYNTAX TABLE LIST
EE03	-4605	PRNTSTR	PRINT A STRING
EE22	-4574	LEN	
EE34	-4556	GETVAL<255	GET VALUE < 255
EE3F	-4545	PLOT	
EE4E	-4530	COLOR	
EE54	-4524	MAN	
EE57	-4521	VTAB	
EE68	-4504	RNGERR	PRINTS '*** RANGE ERR'
EEA0	-4448	CALL	CALL A ML SUBR
EEB0	-4432	HLIN	
EEC6	-4410	VLIN	
EED3	-4397	PRINT	PRINT ERROR MSG/BELL
EEF6	-4362	PEEK	
EF00	-4352	GETVAL255	GET A VALUE FOR 1 BYTE
EF08	-4344	POKE	
EF10	-4336	DIVIDE	
EF1E	-4322	DIMVARB	DIMENSION A VARIABLE
EF4E	-4274	RND	RANDOM # GENERATOR
EFEC	-4116	RUN	RUN FROM BEGINNING
EFF2	-4110	RUN #N	RUN FROM LINE #
F000	-4096	SCRATCH	SCRATCH EVERYTHING
F04D	-4019	HIMEM	
F0C9	-3895	LOMEM	
F0DF	-3873	LOAD	LOAD A PGM FROM TAPE
F11E	-3810	SETHDR	SETUP HDR FOR SAVE/LOAD PARAM
F12C	-3796	SETBUF	SETUP PGM SAVE/LOAD PARAM
F140	-3776	SAVE	SAVE A PGM TO TAPE
F161	-3743	PRNTERR	PRINT AN ERROR MESS
F167	-3737	POP	
F171	-3727	TRACE	
F176	-3722	NOTRACE	
F17D	-3715	TRACEIT	EXEC TRACE FUNC
F279	-3463	STEP	FOR/NEXT STEP FUNC
F2E0	-3360	NODSP	
F304	-3324	DSP	
F30A	-3318	CON	CONTINUE EXEC
F31D	-3299	ASC	
F33B	-3269	PDL	
F351	-3247	RDKEY	READ AN INPUT
F371	-3215	EXP ^	RAISE TO A POWER
F3C9	-3127	PR#S	
F41A	-3046	IN#S	

MICRO™

Turn your Apple into the world's most versatile personal computer.

The SoftCard™ Solution. SoftCard turns your Apple into two computers. A Z-80 and a 6502. By adding a Z-80 microprocessor and CP/M to your Apple, SoftCard turns your Apple into a CP/M based machine. That means you can access the single largest body of microcomputer software in existence. Two computers in one. And, the advantages of both.

Plug and go. The SoftCard system starts with a Z-80 based circuit card. Just plug it into any slot (except 0) of your Apple. No modifications required. SoftCard supports most of your Apple peripherals, and, in 6502-mode, your Apple is still your Apple.

CP/M for your Apple. You get CP/M on disk with the SoftCard package. It's a powerful and simple-to-use operating system. It supports more software than any other microcomputer operating system. And that's the key to the versatility of the SoftCard/Apple.

BASIC included. A powerful tool, BASIC-80 is included in the SoftCard package. Running under CP/M, ANSI Standard BASIC-80 is the most powerful microcomputer BASIC available. It includes extensive disk I/O statements, error trapping, integer variables, 16-digit precision, extensive EDIT commands and string functions, high and low-res Apple graphics, PRINT USING, CHAIN and COMMON, plus many additional commands. And, it's a BASIC you can compile with Microsoft's BASIC Compiler.

More languages. With SoftCard and CP/M, you can add Microsoft's ANSI Standard COBOL, and FORTRAN, or Basic Compiler and Assembly Language Development System. All, more powerful tools for your Apple.

Seeing is believing. See the SoftCard in operation at your Microsoft or Apple dealer. We think you'll agree that the SoftCard turns your Apple into the world's most versatile personal computer.

Complete information? It's at your dealer's now. Or, we'll send it to you and include a dealer list. Write us. Call us. Or, circle the reader service card number below.

SoftCard is a trademark of Microsoft. Apple II and Apple II Plus are registered trademarks of Apple Computer. Z-80 is a registered trademark of Zilog, Inc. CP/M is a registered trademark of Digital Research, Inc.

MICROSOFT
CONSUMER PRODUCTS

Microsoft Consumer Products, 400 108th Ave. N.E., Bellevue, WA 98004. (206) 454-1315

CIRCLE READER SERVICE NUMBER

Atari Error Messages

This program, when included in a BASIC program, will display the English language versions of Atari's number-coded error messages.

David P. Allen
19 Damon Road
Scituate, Massachusetts 02066

Within a few days after putting my Atari computer in operation I wound up with a sore thumb. This was a result of having to continually thumb through the Atari manual to find out the meaning of the latest error message which the Atari was giving me. My threshold of discontent was being depressed lower and lower by the invidious message "ERROR- 12 AT LINE 200", which continually thrust me back to the manual to find out just what I had done wrong. I figured there had to be a better way. There is.

Atari BASIC language is equipped with the very handy 'TRAP' function which you can cause to spring into action every time it encounters an error condition. This command tells the computer to go to the line number immediately following the command (such as TRAP 32000) and continue executing the program at that point. The solution to my problem was simply to direct the computer to a list of error messages with instructions to find the right message, and then print it out on the screen in plain English.

Here's the way it works. The error trapping subroutine is started at line 32500, high enough to be included in most programs without getting in the way of the rest of the program. Way up in the beginning, at the earliest line possible (line 0 is a good place) we enter 'DIM SNAP$(50): TRAP 32500'. This sets up SNAP$ to collect the error messages ('snap', 'trap', — get it? Oh, well...) and instructs the program to

```
2  PRINT "}": REM  CLEARS SCREEN
5  DIM SNAP$(50):TRAP32500
6  REM
7  REM
10  REM  <<< ERROR TRAPPING DEMO >>>
20  REM  <<<  BY DAVID P. ALLEN  >>>
30  REM
40  REM
50  REM
60  REM
70  REM
80  REM  THIS PROGRAM DEMONSTRATES
90  REM  THE PRINTOUT OF ERROR
100  REM  STATEMENTS. THE FOLLOWING
110  REM  LINES ARE DESIGNED TO PRO-
120  REM  DUCE ERRORS. AFTER EACH
130  REM  ERROR, TYPE 'GOTO' PLUS THE
140  REM  THE LINE NUMBER WHERE THE
150  REM  ERROR OCCURRED +10. I.E.,
160  REM  IF THE ERROR MESSAGE SAYS
170  REM  THE ERROR OCCURRED AT LINE
180  REM   220, THEY TYPE 'GOTO 230' TO
190  REM  CONTINUE THE DEMONSTRATION
195  REM
197  REM
200  GOTO 1000
210  NEXT X
220  READ Y
230  SAVE "D2:TEST"
240  PRINT #1,A$
250  PRINT "}": POSITION 5,12
260  PRINT "*** END OF DEMONSTRATION ***"

270  END
326  LIST 32660
32490  REM  <<< ERROR TRAPPING >>>
32491  REM  <<<   SUBROUTINE   >>>
32493  REM
32494  REM
32495  REM  INSERT 'DIM SNAP$(50):
32496  REM  TRAP 32500' AT AN
32497  REM  EARLY LINE NUMBER.
32498  REM
32499  REM
32500 SNAP =  PEEK (195):LNM = 256 *  PEE
K (187) +  PEEK (186): GOSUB SNAP + 32500:
 PRINT "*** ":SNAP$: PRINT "AT LINE ";LNM;"
 ***"
32501 TRAP32500: PRINT " ": END
32502 SNAP$ = "INSUFFICIENT MEMORY": RETU
RN
32503 SNAP$ = "VALUE ERROR": RETURN
32504 SNAP$ = "TOO MANY VARIABLES": RETUR
N
32505 SNAP$ = "STRING LENGTH ERROR": RETU
RN
```

proceed at line 32500 whenever it encounters an error condition. Line 32500 takes a PEEK at two locations which find out first what error occurred (SNAP), and where it occurred (LNM). The computer then finds the correct error message and prints it out on the screen.

Line 32501 resets the trap and ends the program, but you can have your program continue. If you replace 'END' with 'INPUT A$: GOTO LNM + 10' your program will pause at the error message while you reflect on the wisdom of what it is telling you, then when you press 'RETURN' the program will jump to the line number that is ten places further down from where the error occurred. To make this work, all your line numbers must be ten numbers apart, and you must 'DIM A$(1)' back in the beginning of the program. If you leave 'END' in place in line 32501, then you must use 'RUN' or some other immediate command to get things going again.

To save this program for inclusion in your future programs, enter lines 32500 through 32761 into memory through your keyboard. If you are going to save the routine on cassette, then set the program recorder up to record and execute 'LIST "C:" ' and the whole nine yards will be saved on your tape in tokenized form. To retrieve it for use in another program execute 'ENTER "C:" ' after cuing up your tape to the right spot for this routine. The error trapping subroutine will then be added to whatever program you have in BASIC memory at that time.

Disk users follow almost the same routine except use 'D:' and a filename where 'C:' is mentioned above. The filename will be the one you use to identify this subroutine on your disk. I use 'ERRSUB.LST' which reminds me that this file was put on the disk with a 'LIST' instead of a 'SAVE'.

That's all there is to it. If you enter the listing contained herein, the line numbers below 32490 will cause a demonstration of the subroutine procedure to be executed. The price you pay for all of this is the use of 1982 bytes of memory. Atari 800 users with 48K of RAM memory will not give this a second thought; Atari 400 users with only 8K will pause and reflect before dedicating almost 2K to the reduction of their irritation. If it fits your program and your memory then try it out. You'll like it.

MICRO™

```
32506 SNAP$ = "OUT OF DATA": RETURN
32507 SNAP$ = "ERROR > 32767": RETURN
32508 SNAP$ = "INPUT STATEMENT ERROR": RE
TURN
32509 SNAP$ = "DIM ERROR": RETURN
32510 SNAP$ = "ARGUMENT STACK OVERFLOW":
RETURN
32511 SNAP$ = "FP OVER/UNDERFLOW ERROR":
RETURN
32512 SNAP$ = "LINE NOT FOUND": RETURN
32513 SNAP$ = "NEXT WITHOUT FOR": RETURN

32514 SNAP$ = "LINE TOO LONG": RETURN
32515 SNAP$ = "GOSUB/FOR LINE DELETED": R
ETURN
32516 SNAP$ = "RETURN WITHOUT GOSUB": RET
URN
32517 SNAP$ = "GARBAGE": RETURN
32518 SNAP$ = "INVALID STRING CHARACTER":
 RETURN
32519 SNAP$ = "CAN'T LOAD - TOO LONG": RE
TURN
32520 SNAP$ = "DEVICE # >7 OR =0": RETURN

32521 SNAP$ = "NON-LOAD FILE": RETURN
32628 SNAP$ = "BREAK KEY ABORT": RETURN
32629 SNAP$ = "IOCB ALREADY OPEN": RETURN

32630 SNAP$ = "NON-EXISTENT DEVICE": RETU
RN
32631 SNAP$ = "IOCB WRITE ONLY": RETURN
32632 SNAP$ = "INVALID COMMAND": RETURN
32633 SNAP$ = "DEVICE/FILE NOT OPENED": R
ETURN
32634 SNAP$ = "ILLEGAL IOCB #": RETURN
32635 SNAP$ = "IOCB READ ONLY": RETURN
32636 SNAP$ = "END OF FILE": RETURN
32637 SNAP$ = "RECORD > 256 CHARACTERS":
RETURN
32638 SNAP$ = "DEVICE DOESN'T RESOND": RE
TURN
32639 SNAP$ = "GARBAGE AT SERIAL PORT": R
ETURN
32640 SNAP$ = "SERIAL BUS INPUT FRAMING E
RROR": RETURN
32641 SNAP$ = "CURSOR OUT OF RANGE": RETU
RN
32642 SNAP$ = "SERIAL BUS DATA FRAME OVER
RUN": RETURN
32643 SNAP$ = "SERIAL BUS DATA CHECKSUM E
RROR": RETURN
32644 SNAP$ = "WRITE PROTECTED": RETURN
32645 SNAP$ = "DISK/SCREEN MODE HANDLER E
RROR": RETURN
32646 SNAP$ = "FUNCTION NOT IMPLEMENTED":
 RETURN
32647 SNAP$ = "GRAPHICS MODE NEEDS MORE M
EMORY": RETURN
32660 SNAP$ = "DRIVE NUMBER ERROR": RETUR
N
32661 SNAP$ = "DISK FULL": RETURN
32662 SNAP$ = "DISK FULL": RETURN
32663 SNAP$ = "UNRECOVERABLE SYSTEM DATA
- I/O ERROR": RETURN
32664 SNAP$ = "FILE NUMBER MISMATCH": RET
URN
32665 SNAP$ = "FILENAME ERROR": RETURN
32666 SNAP$ = "POINT DATA LENGTH ERROR":
RETURN
32667 SNAP$ = "FILE LOCKED": RETURN
32668 SNAP$ = "INVALID COMMAND": RETURN
32669 SNAP$ = "DIRECTORY FULL": RETURN
32670 SNAP$ = "FILE NOT FOUND": RETURN
32671 SNAP$ = "POINT INVALID": RETURN
```

Introduction to OS-65D V3.3

OS-65 V3.2 was an ultra sophisticated development-oriented operating system. However, several problems kept arising:

1. Output was difficult to format in BASIC.
2. There was no way to trap disk errors in BASIC.
3. Disk file operations were both slow and limited.
4. The nature of the OSI polled keyboard made the use of lower case alphabetics tedious.

OS-65D V3.3 has been designed to eliminate these problems in earlier releases of 65D. In addition, the 65D BASIC line editor has been added as a permanent feature of BASIC. The following describes all the changes that have been made in V3.3. Enjoy!

Compatability

OS-65D V3.3 has the BASIC workspace moved to $3A7E as opposed to $327E on OS-65D V3.2. This change makes no difference whatsoever to the average BASIC programmer. In fact, enhancements to 65D V3.3 allow existing V3.2 files to be both upward and downward compatible to the new system. However, care must be taken when using V3.2 files that contain assembler language subroutines. The subroutines will be transferred, along with the program that contains them, but will be physically relocated in memory and will probably not execute properly, if at all.

Programmable Error Action

In OS-65D V3.3 BASIC, the WAIT command has been replaced by the TRAP function which is used as an "ON ERROR GOTO" (but is easier to type). The TRAP function can be used either in the immediate mode or inside BASIC program and is effective whether a BASIC ERROR or DOS ERROR occurred. For example, consider the following program segment:

```
10   TRAP 1000
20   DISK OPEN,6,"DATA"
30   TRAP 40
40   INPUT#6,A:B=A/A
50   PRINT A:END
1000 ?"DISK ERROR":END
```

If a DISK ERROR occurred in line 20, control would be transferred to line 1000. Lines 30 and 40 are used to read the first non-zero number in the file. The TRAP function is disabled by the statement "TRAPO".

Keyboard Driver

The standard OSI polled keyboard driver has been replaced in OS-65D V3.3 by an all new keyboard decoder. The SHIFT LOCK key now acts as a CAP LOCK key and the RUBOUT key actually does delete characters. However, three characters still cause problems. These are listed below along with their keystroke equivalents:

Λ - SHIFT N
[- SHIFT K
] - SHIFT M

Note: The SHIFT LOCK key must be depressed when these three characters are typed.

Random Files

OS-65D V3.3 incorporates several improve ments in the random file capabilities in OS-65D. First, the DISK GET command has been altered to check which track is currently resident in RAM before actually reading a track. If the GET command determines that this is the tract that is needed, no reread is performed. Thus, the random file access time is up to 48 times faster than in 65D V3.2.

Secondly, a DISK FIND command has been added. The syntax is "DISK FIND,string" where string is any BASIC string variable or quoted literal. The search begins at the current file pointer and will continue through the file. If the string is not found, an ERR#D will be reported (unless the TRAP command is used). If the string is found, the file pointer will be set to the beginning of the *next* field entry. For example:

```
10  DISK OPEN,6,"DATA"
20  DISK GET,10
30  PRINT#6,"HELLO":PRINT#6,"THERE!"
40  DISK GET,0
50  DISK FIND, "HELLO"
60  INPUT#6,A$
70  PRINT A$
```

This program will print out "HELLO!".

Note: The search rate for the FIND command is about 8K/second on 8" systems and 5K/second on mini-floppies.

Printer Drivers

The printer drivers in OS-65D V3.3 (devices 1 and 4) have a programmable paging feature which is enabled by the following:

```
PRINT#LP,CHR$(27);"C";CHR$(FL)
```

where LP is the printer device number and FL is the form length you want. Ten percent of the form length is always reserved for the top and bottom margins. For example,

```
PRINT#1,CHR$(27);"C";CHR$(66)
```

indicates form length of 66 where 60 lines are printed per page and six lines are reserved for the top and bottom margins. Immediately after the form length is set, a top of form is executed. At this time, position the paper in the printer as desired. To print a top of form to the next page, enter

```
PRINT#LP,CHR$(12);
```

The printer drivers also have a screen dump utility which may be used if you have an EPSON MX-80 printer *and* a standard OSI 540 video system. To use this feature, enter

```
PRINT#LP,CHR$(27);"P";
```

OS-65D V3.3 Editor

In OS-65D BASIC, the keyword NULL is replaced by the word EDIT. After the system is booted, immediately type a non-destructive forward and backspace to tell the editor what type of keyboard you are using, (CTRL-L and CTRL-P are the forward space and backspace, respectively, for the OSI keyboards.) The syntax for editing a line is given in table 1.

Table 1

0 =< LN < 64000

EDIT LN < CR > or !LN < CR >	Edit the statement with the line number LN.
EDIT! < CR > or !! < CR >	Edit the same line that was just edited.
EDIT < CR > or ! < CR >	Edit the line immediately following the line that was just edited.

The line with its line number will be displayed following the <CR>. If the line number LN does not exist, the statement with the next line number will be displayed. (Typing EDIT0 or !0 will always give the first line of the program.) After the statement is displayed, the cursor will reside at the end of that line. The commands listed in table 2 are used for the actual line editing.

Table 2

→ /CTRL-L/Forward Space	Non-destructive forward space. Moves the cursor one space to the right.
← /CTRL-H/CTRL-P/Backspace	Non-destructive backspace. Moves the cursor one space to the left.
RUBOUT/DELETE/SHIFT-0	Single character delete. The editor makes the correct delete keys operational as well as the old ones (i.e., the RUBOUT key as well as SHIFT-0 will work on the OSI polled keyboard when the editor is enabled).
@/SHIFT-P	Entry delete. This will erase the line currently being edited, leaving the line in the text as it was before it was edited.
CTRL-R	Non-destructively moves the cursor to the rear of the statement.
CTRL-F	Non-destructively moves the cursor to the front of the statement.
CTRL-I	Non-destructively moves the cursor eight spaces forward (to the right).
CTRL-T	Retypes the statement you are currently editing.
< CR > / < RETURN > / < ENTER >	Enters the line as written or viewed. The line will look (to the BASIC interpreter) as if it were typed in by the user from scratch.

Character insertion and deletions can be accomplished anywhere by using the commands for non-destructive movement of the cursor. After the cursor is positioned, the user can type in insertions or delete unwanted characters. *Note:* Characters are inserted to the left of the character on which the cursor resided. The character on which the cursor resides is deleted until the end of the line is reached, and the characters to the left will be deleted if the cursor resides at the end of a line.

Video Driver

The video driver for 65D has been rewritten in order to provide (X,Y) cursor addressing and more than a dozen screen editing commands. These commands are used by printing CHR$(27), an ASCII ESC, followed by the desired command. For example,

```
PRINT CHR$(27);CHR$(28);
```

clears the video screen and homes the cursor. The rest of the commands are given in table 3.

Table 3

Code	Effect
CHR$(1)	Causes following data to be printed in the color yellow.
CHR$(2);CHR$(n); CHR$(m)	All screen positions marked by color n are changed to color m.
CHR$(5)	Sends the current cursor address through the keyboard driver, i.e., PRINT CHR$(27); CHR$(5);:INPUT A$ then, A$ = CHR$(65 + X) CHR$(65 + Y)
CHR$(11)	Cursor moves down one line.
CHR$(12)	Cursor moves up one line.
CHR$(15)	Clears from the current cursor position to the end of line.
CHR$(17);CHR$(X): CHR$(Y)	Moves cursor to screen position (X,Y). $0 \leq X < 64$ $0 \leq Y < 24$
CHR$(18)	Moves cursor to the home position, i.e., (0,0) — the upper left-hand corner.
CHR$(19)	Deletes the line the cursor is on. Lines below the cursor scroll up one line.
CHR$(24)	Clears from the current cursor position to the end of the screen.
CHR$(25)	Causes output to be printed in no color (black).
CHR$(26)	Inserts a line at the cursor position. Lines below the cursor scroll down one line.
CHR$(28)	Clears screen and homes cursor.
CHR$(29);CHR$(n)	Clears all occurrences of color n on the screen.
CHR$(31);CHR$(n)	Causes the following data to be printed in the color n.
CHR$(33)	Sends the character at the cursor position through the keyboard driver, e.g., Print CHR$(27); CHR$(33) INPUT A$

Indirect File Problems? Why Not Use a Diskette?

In this section we describe a method for merging two BASIC files under the OS-65D operating system. The procedure uses the disk I/O capabilities of 65D to make your diskette into an indirect file. The following step-by-step procedure can be used to merge two programs. We start with both programs, say PROG1 and PROG2, stored on a diskette.

PROG1 N1 = 1 TRACK

```
40  REM THIS IS PROG1
50  REM
60  REM
70  END
```

PROG2 N2 = 1 TRACK

```
10  REM THIS IS PROG1
20  REM
30  REM
40  END
```

1. Load PROG 1 into the workspace

DISK!"LOAD PROG1"

Enter

EXIT

The number of tracks necessary to hold PROG1 will be displayed, say N1 tracks. Return to BASIC by entering

RE BA

Now do the same with PROG2, obtaining its size, N2 tracks.

2. Run the disk utility CREATE and create a file PROG3, N1 + N2 (N3) tracks long, to hold the merged programs. If PROG2 already has enough space, the merged program can be stored as PROG2.

3. The number, N1, of tracks necessary to store PROG1 was determined in step 1. Run CREATE again and make a file called "DATA" with three times N1 tracks for a five inch diskette, and four times N1 tracks for an eight inch diskette. Answer NO to the query about pages per track. Specify four pages per track.

4. Load PROG1 into the workspace

DISK!"LOAD PROG1"

5. Enter the following POKEs to create a four-page buffer and to disable the scrolling of the screen (the screen will hold the buffer).

```
POKE  8998,0
POKE  8999,208
POKE  9000,0
POKE  9001,212
POKE  9770,0
```

6. Enter on a *single line*

DISK OPEN,6,"DATA":DISK!"IO ,22":LIST

A listing of the workspace will appear on the screen while PROG1 is being stored in the file DATA.

7. When the listing is finished, reset the I/O pointers and close the file by entering

DISK!"IO 02,02":DISK CLOSE,6

8. Load PROG2 into the workspace by entering

DISK!"LOAD PROG2"

9. Reopen the file DATA and merge PROG1 into PROG2 by entering

DISK OPEN,6,"DATA":DISK!"IO 20"

10. Reset the I/O pointers, close the file, and enable scrolling by entering

DISK!IO 02,02":DISK CLOSE,6
POKE 9770,64

11. Store the merged file by entering

DISK!"PUT PROG3"

12. Clean house by rebooting the system.

If each of the programs has a line with the same number, the line in PROG1 will be the one that appears in the merged program.

MERGED PROGRAM PROG3 N3 = 1 TRACK
N1 + N2

```
10  REM THIS IS PROG2
20  REM
30  REM
40  REM THIS IS PROG1
50  REM
60  REM
70  END
```

Note: Line 40 of PROG2 was overwritten.

Finally, by changing the LIST specification in step 6, you can merge any part of a program, or just break up large programs. The uses are unlimited.

OHIO SCIENTIFIC 1333 S. Chillicothe Road • Aurora, Ohio 44402 • (216) 831-5600

DISK 1

A fantastic new game disk for OSI-C4P's running with DOS 3.2. Most of the games on this disk are single player such as a multiple level Othello game, Cavern Chase, and more. Other games are designed for single or multiple players such as Tank, and Crystalize. Over seven games and variations in all. (Note that many of these games are in machine code for enhanced play-ability and speed.)

To order send **$26** *to:*

Simulations Programming
Rt. 2 Box 98
Burton, WA
98013

SOFTWARE FOR OSI

VIDEO GAMES 1 .... $15.
Three Games. Head-On is like the popular arcade game. Tank Battle is a tank game for two to four. Trap! is an enhanced blockade style game.

VIDEO GAMES 2 .... $15.
Three games. Gremlin Hunt is an arcade-style game for one to three. Gunfight is a duel of mobile artillery. Indy is a race game for one or two.

ADVENTURE: MAROONED IN SPACE .... $12.
An adventure that runs in 8K! Save your ship and yourself from destruction.

DUNGEON CHASE .... $10.
A real-time video game where you explore a twenty level dungeon.

BOARD GAMES 1 .... $15.
Two games. Mini-gomoku is a machine language version of five stones gomoku. Cubic is a 3-D tic-tac-toe game. Both with graphics.

DISASSEMBLER .... $12.
Use this to look at the ROMs in your machine to see what makes BASIC tick. Reconstruct the assembler source code of machine language programs to understand how they work. Our disassembler outputs unique suffixes which identify the addressing mode being used, no other program has this!

SUPER! BIORHYTHMS .... $15.
A sophisticated biorhythm program with many unique features.

C1 SHORTHAND .... $12.
Use only two keys to enter any one of the BASIC commands or keywords. Saves much typing when entering programs. Written in machine language.

For all BASIC-in-ROM systems. Selected programs available on disk. Color and sound on video games

Send for **FREE** catalog

ORION SOFTWARE ASSOC.
147 Main St. Ossining, NY 10562

OSI BASIC

Summary Manual for 65U & 65D
8" Disk – 65U V1.2 Utility Enhancements

Tired of searching for commands & statements, peeks, error msgs, system limits? Need hints? Then send $9 for our 12/80 manual.

Want prompts for utility pgm inputs? File searching demo? pgm exits? round numbers to 3 decimals? more disk space with utility pgms? latest updating? Then send license copy and $20 for 8" Floppy. Specify video or serial.

Both for license copy and $25.

(Indiana residents add 4% sales tax.)

C & J Supply
Box 806, Marion, IN 46952

"The ROM Rabbit"

ROM and MANUAL — $49.95
Specify 3.0 or 4.0 Basic (Works with or without toolkit)

EASTERN HOUSE SOFTWARE
3239 Linda Drive
Winston-Salem, N. C. 27106
(919) 924-2889
(919) 748-8446

MICRO Challenges

By Paul Geffen

The Superboard

The OSI Model 600, better known as the Superboard, is one of the oldest of the single-board microcomputers. At $299 (list) it provides more computing power per dollar than almost any other system in its class. For the beginner this system is close to ideal, both affordable and accessible.

For $299 you get one board with these features: a typewriter-style keyboard, a 6502 microprocessor, 4K of program RAM, 1K of display RAM, video output circuitry, 8K Microsoft BASIC-in-ROM and a 2K ROM monitor. The board comes with a User's Manual to help the beginner find his way around. You will also need a power supply (five volts at two amps) and a video monitor or an RF modulator to connect the computer to a TV set. These may cost between fifty and two hundred dollars more.

Or you could buy the OSI C1P for $429, which is a Model 600 with an extra 4K of program RAM in a case with a power supply. A good cassette recorder (not battery operated) is almost essential, and you would do well to arm yourself with additional reference materials which I will describe below.

Now you have a complete computer system which will allow you to write programs in BASIC and/or machine language. You can also run programs which others have written, as long as they were written for the OSI Superboard or C1P. Converting BASIC programs from other machines is sometimes easy, but sometimes almost impossible. For instance, tapes written for other micros probably won't load on the OSI.

Information Resources

As they become more experienced, most beginners notice that there is a lot that can be done with the Superboard that isn't explained or even hinted at in the OSI documentation. Many of the apparent limitations of the board are really only deficiencies in the User's Manual. Of course this is what user's groups are for. There are a few good books available which offer much useful information, both for the beginner and the experienced programmer.

Perhaps the best to start with is Ed Carlson's *OSI BASIC in ROM*. This book, now in its second edition, is written by a C2P/C4P user, but almost everything in the book applies to the Superboard because the same BASIC comes with both machines. Carlson describes the capabilities of BASIC in considerably more detail than the User's Manual and he includes a few things the manual leaves out, like the bugs. Carlson goes into detail on solutions to the infamous Garbage Collector bug (which OSI doesn't even mention). Then he explains how to write good, well-organized BASIC programs and he provides many useful utility programs for clearing the screen (fast), converting hex to decimal, writing monitor format tapes, and so on.

The material on the actual mechanics of the BASIC interpreter is very useful and informative, and gives the key to many clever and efficient ways of writing programs for this machine. Finally, this book has a comprehensive list of publications and vendors of software for the OSI. In short, this is the book every Superboard owner should have alongside his User's Manual.

The next book I recommend for the more experienced user, is Williams and Dorner's *First Book of OSI*, published by Aardvark Technical Services. (Do not confuse this with a book with almost the same title, by Clothier and Adams, published by ELCOMP.) Williams and Dorner provide much of the same information as Carlson, with less introductory material, and more technically sophisticated material. Williams and Dorner's book is not for the beginner, as Carlson's is, so this should be the second book to buy.

The User's Manual does have some good points, namely a good job of printing and production. Of the books I have mentioned, it has the best graphics table and the best hex-to-decimal conversion table. Carlson has the most complete map of BASIC ROM entry points. Williams and Dorner go into more detail on what the ROM routines do.

The *First Book of Ohio Scientific* and *Second Book of Ohio Scientific* by Clothier and Adams contain mostly promotional material reprinted from OSI newsletters and entries from the "Small Systems Journal." Most of the informative material here is also available, and better presented in Carlson's or Williams and Dorner's books.

Unfortunately, that about covers the available books written specifically for the OSI user. In addition to these books, two other sources of information exist: periodicals and plans for sale.

Two of the best periodicals were described last month, the *OSIO Newsletter* and *PEEK(65)*. I also recommend the *Aardvark Journal*, which is now about six issues old. This is a bimonthly journal, published by a leading supplier of software for OSI systems. For more information write to: Aardvark Journal, 1690 Bolton, Walled Lake, Michigan 48088.

There is one other OSI-only publication, the *Independent Newsletter, O. S. I. U. N.* put out by Charles Curley at 6061 Lime Ave., #2, Long Beach, California 90806. I have seen only one issue of this, and I don't feel it is enough to judge this relatively young newsletter.

MICRO publishes at least one OSI-related article each month as well as this column. *COMPUTE!* has an "OSI Gazette" and *Kilobaud Microcomputing* runs about four or five OSI-related articles per year.

All of these publications supply short programs and hardware projects of real utility as well as good introductory material.

Finally, it is possible to buy plans and/or kits for various modifications to the Superboard. Ads for these run in MICRO and other publications, and similar plans can be found in the above-mentioned publications. For instance, Video Mods, to increase the number of characters displayed per line, are described in Aardvark #5 (simple), and *PEEK(65)* #11 (complex).

I plan to publish a more complete list of OSI information resources in a future column. I am sure that I have overlooked some newsletters and magazines. I am particularly interested in boards or kits or plans that will: expand the Superboard memory, expand the video display, increase the cassette speed, and provide RS 232 and modem support. Please send catalogs, etc. in care of this column, to ensure your product's inclusion in this list.

BASIC Program Converter Between SYM and KIM

This program allows a person to transfer BASIC programs from SYM to KIM or from KIM to SYM without having to spend hours typing in and debugging the programs.

Lee Chapel
2349 Wiggins
Springfield, Illinois 62704

Have you ever wanted to put somebody's SYM BASIC program on your KIM without spending hours typing the program in and debugging it? Or have you ever wanted to put somebody's KIM program on your SYM? This converter program allows you to easily transfer BASIC programs from SYM to KIM, or from KIM to SYM. It is especially useful for long, 8 to 12K, programs. I used it to transfer a 14K program I call "Monster Combat" from KIM to SYM in roughly half an hour.

Description of BASIC Format

The BASIC format used in the SYM and KIM is as follows. The first two bytes of a program line point to the start of the next line (see diagram). The next two bytes are the line number, and the remaining bytes are BASIC tokens or data in ASCII. A token is one byte (80 to C5 hex) which represents a 2 to 6 letter BASIC word.

Tokens or Data in ASCII Code (Hex)

LO HI Byte Next line pointer in hex	LO HI Byte Line number in hex

The tokens in both SYM and KIM (such as INPUT and PRINT) are the same hex value. For example, on both systems INPUT is 84 in hex, and PRINT is 97 in hex. An example of how a BASIC line is formed is shown in figure 1.

Figure 1

4000	00 14	40 05	00 97	22	48	49	20	54	48	45	52	45	22	3A	88	35
	Pointer to next line	Line number	PR INT	"	H	I		T	H	E	R	E	"	:	GO TO	5

Comparison of SYM and KIM Systems

KIM begins program storage at 4000 hex, SYM begins storage at 0200 hex. Since the data and the tokens are the same, only the line pointers and actual program location in memory need to be changed. The program can be relocated on SYM by use of the Block Move, "B". On KIM the use of a supplementary monitor such as "XIM" can be used to relocate the program. It's also possible to relocate the program by using the tape load FF function and new address. The regular KIM tape record and playback are the same as the low speed SYM tape record and playback.

The only remaining difference between the two systems is the pointer values. They all need to be changed to reflect the new location in the other system. The BASIC converter program is written to convert all these pointer locations. The BASIC program takes only a few seconds to convert long programs, so speed is not a problem.

Converter Program Description

In both program listings, A is the address where the low byte of the first pointer is located. B is the value found in the address A, and C is the value of the high byte of the pointer. D is set equal to the first hex digit of C, and E is set equal to the other hex digit of C. D, E, and B are then placed in an equation where F becomes the value of the address of the next line pointer. Since only the high byte needs to be changed, the address A + 1 is POKEd with a new value. A is then set equal to F and the entire process continues with a new value of the line pointer until two zeros are found in adjacent addresses.

Program Examples

The following is an example of a KIM to SYM conversion. First check addresses 7D and 7E. These are, respectively, the low and the high bytes of the end of the program being transferred. Make a record of these values and make a recording from 4000 hex to the address in those two memory locations at normal record speed. Next the tape is loaded into SYM at slow speed and placed in memory so that it starts at 4000. SYM BASIC is then entered with a J 0 and when Memory Size is asked for, a low value, such as 1500, should be entered. Type in the converter program. Make sure there are no errors and then run the program. Once the program finishes, go back into the monitor, move the program at 4000 hex down to 0200 hex. Take the value in 7D that you noted and subtract 3E hex

from it, and place that number in 7D. Next take the value noted for 7E and place it in 7E. Then set memory locations 87 and 88 to the proper size of your BASIC program area.

Converting from SYM to KIM is similar. Again, check memories 7D and 7E and make a note of them. Make a tape of the program in the SYM's low speed format. Load it into KIM and place it in memory so that it starts at 5000 hex. Then start KIM BASIC and when you are asked for Memory Size, give a low value such as 17000. Type in the conversion program, make sure there are no errors, and run it. When the program finishes, go back to the monitor and move the converted program from 5000 to 4000. Take the value noted for 7D and add 3E hex to it. Take the value noted for 7E, and place it in 7E. Change memory locations 87 and 88 to the proper size of your BASIC program memory. After moving and testing, a final tape dump can be made.

Conclusion

I have used both these conversion programs successfully on several BASIC programs. Any USRs or special I/Os will have to be modified. It should also work on AIM, assuming the tokens are the same. These programs have saved many hours of retyping programs between systems.

Lee Chapel has been working with KIM and SYM for about 3 years. He is majoring in Computer Science at the University of Wisconsin-Madison and has worked there as a programmer in the Agriculture Economics department.

MICRO

Listing 1

```
5  REM  SYM TO KIM CONVERSION IN BASIC
10 A=20481
15 IFPEEK(A)=0ANDPEEK(A+1)=0THENEND
20 B=PEEK(A):C=PEEK(A+1)
25 D=INT(D/16):E=C-16*D
30 F=4096*D+256*E+B+19968
35 POKEA+1,C+62
40 A=F:GOTO15
45 END
```

Listing 2

```
5  REM  KIM TO SYM CONVERSION IN BASIC
10 A=16385
15 IFPEEK(A)=0AND PEEK(A+1)=0THENEND
20 B=PEEK(A):C=PEEK(A+1)
25 D=INT(C/16):E=C-D*16
30 F=4096*D+256*E+B
35 POKEA+1,C-62
40 A=F:GOTO15
```

MICRO Microbes and Updates

Mike Rowe
Microbes & Updates
P.O. Box 6502
Chelmsford, MA 01824

This month, we offer the following improvements, rather than corrections.

Les Cain, of Grand Junction, Colorado, sent this update to his article in the January issue (32:75): There is a problem with Control C in "Fun with OSI." I apologize to the readers for the oversight in not replacing the Disk control C POKE with the proper ROM POKE. To correct the problem, change the following lines:

```
Line  760 POKE 530,1:K=57088
Line 1710 POKE 530,0:END
Line 1740 POKE 530,0:END
```

Charles F. Taylor, Jr. of Monterey, California, offers the following tip: "Business Dollars and Sense in Applesoft" by Barton M. Bauers, Jr. (MICRO 27:65) was most interesting and useful. Here are a couple of simple changes which will make the routine "Mask" even more useful:

1. "Mask" produces output left-justified in a variable-width field. While this is useful for some applications, it will not do for producing columns of figures. Ideally, the output should be right-justified in a predetermined field width (specified by the user). This can be accomplished by adding lines 16 and 15025 and by modifying line 15110 as shown below:

```
   16 FW = 12 : REM SET FIELD
      WIDTH (5   =FW   =12)
15025 BL$ = "        " : REM 8 BLANKS
15110 XW$ = XV$ + LEFT$(BL$,
      FW-4-LEN(XX$)) + XX$ +
      XZ$
```

Line 16 as shown, produces the maximum allowable field width and is sufficient to handle dollar amounts from $-999,999.99 to $9,999,999.99, which is the range handled by Mr. Bauers' original routine, and which should be adequate for most small applications. (It is certainly adequate for my personal checkbook.) The field width can be changed anywhere in the calling program by assigning the desired value to the variable "FW". Line 15110 as shown left-justifies the leading "$", but this is easily changed.

2. Because of the behavior of the function "INT" (described by Mr. Bauers in his article), the routine as it stands will round fractional cents incorrectly for negative amounts, e.g. -1.009 rounds to -1.00 instead of -1.01. (Fractional cents occur most often when computing percentages.) Here is a simple fix to line 15 which solves the problem:

```
15 DEF FN VL(X) = INT((X +
   SGN(X)*.0001)*100 + .5)
```

John P. Molineaux of Cheverly, Maryland, sent this enhancement: There is always a better way. On reading Frank Chipchase's excellent article on "Better Utilization of Apple Computer Renumber and Merge Program" in the August 1980 issue (27:17), I was struck by the awkwardness of the series of EXEC file POKEs required to configure the A/S-R/N-M program. Machine language is far better than Applesoft at POKEs and it doesn't fill the screen with Applesoft prompt characters (]).

Recall that HIMEM:32352 is equivalent to POKE 115,0:POKE 116,142. Hiding A/S-R/N-M and resetting the &-pointer therefore translates to:

Assembly	**Decimal**
LDA 0	169 0
STA 115	133 115
STA 1014	141 246,3
LDA 142	169 142
STA 116	133 116
STA 1015	141 247,3
LDA 76	169 76
STA 1013	141 245,3
RTS	96

If this short program is appended to the front of the A/S-R/N-M and the whole mess is BSAVEd as a unit under the name RENUM, then the loading and reconfiguration is quickly achieved by

```
BRUN RENUM
```

Here's how:

1. RUN Apple's RENUMBER from the system disk.

2. POKE in the 20 bytes of the program as follows:

```
POKE 36332,169
POKE 36333,0
POKE 36334,133
POKE 36335,115
POKE 36336,141
POKE 36337,246
POKE 36338,3
POKE 36339,169
POKE 36340,142
POKE 36341,133
POKE 36342,116
POKE 36343,141
POKE 36344,247
POKE 36345,3
POKE 36346,169
POKE 36347,76
POKE 36348,141
POKE 36349,245
POKE 36350,3
POKE 36351,96
```

Of course, the monitor is zippier on this kind of task, if you want to enter the hex equivalents of the decimal POKEs in $8DEC through $8DFF.

3. BSAVE RENUM,A36332,L2068

Notice that an additional 20 bytes spill over onto one more track in the RENUM data set.

This way of saving the program saves a second or two on each run and dispenses with the screenful of empty "]" lines that scoot your last display off the screen. After the BRUN RENUM, the 20-byte program is eventually destroyed the next time an Applesoft string is created.

MICRO Software Catalog: XXXI

Mike Rowe
P.O. Box 6502
Chelmsford, MA 01824

Name: **The DI-SECTOR Disk**
System: Apple II, Apple Plus
Memory: 48K
Language: ROM Applesoft, Assembly
Hardware: Disk Drive, optional; printer, language card
Description: A set of seven disk utility programs to help 'DI-SECT' your disks. Now you can easily display and edit any sector of a disk, list sectors used by a file, create a new VTOC, display free and used sectors, create 'EXEC' files easily, print text files on the screen or a printer, and load the language card with the old ROM code.
Price: $20.00 on disk postpaid. Includes both DOS 3.2 and 3.3 versions, and documentation.
Author: **Jeffrey Durham**
Available: **Mike Rowe Productions**
P.O. Box 43504
Tucson, Arizona 85733

Name: **NDE—Package**
System: CBM Commodore
Memory: 32K
Language: BASIC
Hardware: CBM 3032/
CBM 3040/CBM 3022
Description: It is a whole package for handling the results of X-Ray-Examinations and liquid dye examinations of welds, 6 languages (German, Italian, French, English, Spanish and Portuguese), aic implemented.
Copies: Disk
Price: $800.00
Author: **M. Bauer**
Available: **M. Bauer**
Aindorferstr.86
D-8000 Muenchen 2A
West Germany

Name: **0-1. Options**
System: PET
Memory: 8K
Language: BASIC
Hardware: PET/CBM
Description: Options are evaluated. A unique measure of option value is computed and used to compare options for up to three expiration dates and three striking prices. Normal prices for puts or calls may be computed for any assumed situation and tables printed as a function of stock price.
Price: $15.00 for cassette and documentation
Author: **Claud E. Cleeton**
Available: **Claud E. Cleeton**
122-109th Ave., S.E.
Bellevue, Washington
98004

Name: **Star Cruiser**
System: Apple II or Plus
Memory: 32K RAM
Language: 20K of assembly
Hardware: One drive. Either 13 or 16 sector controller.
Description: Simply stated, the ULTIMATE ACTION GAME. If you're looking for fast Hi-Res action with color, sound and action to rival Atari's Galaxian game, then this is it. Make it past the first two levels and the third level will really kill you. Hundreds sold in the first week of sale.
Price: $24.95
Author: **Nasir Gebelli**
Available: **Sirius Software**
1537 Howe Ave.,
Suite 106
Sacramento, California
95825

Name: **Commodity File**
System: Apple II, Apple Plus
Memory: 32K with Applesoft ROM or 48K with Applesoft RAM
Language: Applesoft II
Hardware: Disk II, 132 column printer (optional)
Description: Commodity File stores and retrieves virtually every commodity traded on all Future's exchanges. A self-prompting program allowing the user to enter short/long contracts. Computes gross and net profits/losses, and maintains a running cash balance. Takes into account any amending of cash balances such as new deposits or withdrawals from the account. Instantaneous readouts (CRT or printer) of contracts on file, cash balances, P/L statement. Includes color bar graphs depicting cumulative and individual transactions. Also includes routine to proofread contracts before filing.
Price: $19.95 plus $2.00 (p&h) first class mail.
Author: **Mind Machine, Inc.**
Available: **Mind Machine, Inc.**
31 Woodhollow Lane
Huntington, New York
11743

Name: **SEX-O-SCOPE™: THE PRIVATE-LIFE HOROSCOPE™**
System: Apple II or TRS-80
Memory: 32K for screen version, 48K for printout version
Language: Apple II—Applesoft in ROM with DOS 3.2; TRS-80—Disk BASIC 2.3
Hardware: Apple II—1 disk with screen version, 2 disks with printout version. TRS-80—2 disks with both versions.
Description: Casts an accurate horoscope, then interprets it from a sexual viewpoint in 1500 words or more. Not just a paragraph about your sign, but rather a reading of all the planets, signs and houses in a horoscope that is unique to the individual. Text, by best-selling author of *Planets In Love* and former editor of *Sexology Today*, outlines tastes, turn-ons and hang-ups in a tolerant, witty style. Computations are precise within a tenth of a degree for any date and time from 1880 to 2000.
Price: $30.00—screen version
$200.00—printout version (includes license to reproduce textual material commercially)
Author: **John Townley and AGS Software**
Available: **AGS Software**
Box 28
Orleans,
Massachusetts 02653

Name: **The Ultimate Catalog**
System: Apple II/Apple II Plus
Memory: Min. 20K (ROM Applesoft)
Language: Applesoft and Machine RWTS
Hardware: Apple II, Disk II, DOS 3.2
Description: Now you can format your directory to appear any way you wish. Block similar programs together; write

headers mid-directory; separate by sections. This 5K, menu-driven utility is easy to use and performs the following functions: Alphabetize any portion or all of directory, move any file, exchange any two files, highlight or remove highlighting from any file name, insert blank line(s), delete any file, lock or unlock all files, delete or restore all files.

Price: $6.50 for listing/instructions
Author: **Larry Abrams**
Available: **ARIES SOFTWARE**
P.O. Box 58
Los Altos, California 94022

Name: **The Math Machine**
System: Apple
Memory: 32K
Language: Applesoft in ROM
Hardware: Disk, optional printer
Description: Kid-tested, effective instructional software to improve math skills. Covers pre-math through division with over 110 skill levels. Designed by educators and written by programmers for use by parents and teachers. Includes such features as: reinforcement system, management, record keeping, individualization, personalized lessons, performance objectives, and immediate feedback.
Price: $79.95 includes original and back-up disk, teacher and parent manual, support.
Author: **Larry Johnson**
Available: **SouthWest EdPsych Services**
P.O. Box 1870
Phoenix, Arizona 85001

Name: **The Mailing Label and Filing System**
System: Apple II
Memory: 48K
Language: Applesoft
Hardware: Apple II, Disk II (one or two drives)
Description: Has loads of features: binary sorting; 1 - 3-second access of records by name or record number; user formatted; optional 9-digit zip code update; performs COUNT/SORTS which enables the user to display a certain type of population off the disk and/or make print-outs or mailing labels; reversible directory reading; special backup programs, insert programs, copy-by-record (for backup) programs; automatic formatting file and directory updating; edit; delete; write; count; read; print; customized mailing labels, quicksort utilization; much more.
Price: $24.95 includes disk, manual, demo sheet.
Author: **Avant-Garde Creations**
Available: **Avant-Garde Creations**
P.O. Box 30161 MCC
Eugene, Oregon 97403

Name: **Z-Term**
System: Apple II or Apple II Plus
Memory: 48K
Language: CP/M, (Machine Language)
Hardware: Disk II (DOS 3.2 or 3.3), Z-80 Softcard, Micromodem II or most other communications devices, 80-column board or external terminal optional.
Description: Upload and download data files between the Apple and another computer. A number of "luxury" features are also provided. Commonly used systems can be put into a directory for auto-dialing, keyboard Macros allow you to define strings for output with simple keystrokes for fast log-ins to system, or to issue various commands within the system. Table-driven prefix keys allow you to produce any character not already on the Apple keyboard without losing any other keys! Fully compatable with standard CP/M sequential text files, and can send files from disk of *any* size, and can receive up to 36K of data at a time. Auto-save mode will send XOFF character to host, save file (with operator prompting) and then continue.
Price: $79.95, introductory (until May 1, 1981)
Author: **Bill Blue**
Available: **Southwestern Data Systems**
P.O. Box 582-M
Santee, California 92071

Name: **Display-it**
System: OSI, C1P or Superboard
Memory: 4K
Language: 8K BASIC-in-ROM
Description: Displays messages from right to left in large letters made up of any graphics character. Messages can be up to 255 characters long.
Price: $5.95 ppd. includes cassette and documentation
Author: **Brian and Craig Zupke**
Available: **BC Software**
9425 Victoria Drive
Upper Marlboro
Maryland 20870

Name: **Small Business Accounting (SBA)**
System: OSI C4P MF
Language: BASIC under OS65D
Hardware: Printer, 2 Disks (second optional)
Description: Provides double-entry journal system for cash flow analysis and reports. Automatic checking of distribution account totals at time of entry. User-defined fields in data base files; up to 99 expense and income accounts, 999 vendor/customer accounts, with names up to 72 characters. Six digit (XXXX.xx) capability in base module is expandable. Prints Income Statement, Trial Balance, Charts of Accounts and Vendor/Customer lists. Summary financial information totalable by month, quarter, and YTD. Sorting is available on user specified fields. All records are MDMS compatible and code allows user system configuration.
Price: $100.00 (3rd Class mail free, 1st Class add $2.00).
Includes: (1) program disk and (1) data disk with sample file. User manual and Accounting System Guide and sample source documents provided. Program listings only are available for $20.00 each.
Author: **J.O. Rector**
Available: **Video Ventures**
1708 Beechwood Avenue
Fullerton, CA 92635

Name: **GRAFPAK APPLE**
System: Apple II
Memory: 32K minimum
Language: Either BASIC and 6502 ML
Hardware: Disk and graphic printer
Description: GRAFPAK is a family of programs for reproducing the Hi-Res pages using grab-the-wire printer graphics. 1× and 2× scaling are standard, and 3× and 5× are available with some printers. Normal and inverse inking is selectable, and variable indent is provided. Features vary with make and model of printer. Packages available for IDS-440,445 and 460, Anadex DP-9xxx family, and Epson MX-70 and MX-80 with graphic PROMs.
Price: $24.95 (+1.65 in Ohio) includes diskette and user's guide (specify DOS release and printer model).
Available: **SmartWare**
2281 Cobble Stone Court
Dayton, Ohio 45431


MR. RAINBOW
presents our valuable free catalog (over 100 pages). He PROMPTS you to PEEK at the latest collection of software and hardware products for your APPLE II™
A STELLAR TREK
the definitive Hi-Res color version of the classic Startrek game. Three different Klingon opponents. Many command prerogatives from use of weapons to repair of damages. Needs 48K Applesoft ROM.
Disk...$24.95
VERSAWRITER II
A drawing tablet, simply plugs into your game I/O port. Trace, draw, design, or color any type of graphic. Adds words to pictures. Creates schematics. Computes Distance/Area of any figure. New - fill any area on the screen in seconds with over 100 different and distinct colors. Needs 32K Applesoft ROM and disk drive. A bargain at...
$249.95
BOWLING DATA SYSTEM
This data mangement program provides accurate record keeping and report generation for bowling leagues of up to 40 teams with 6 bowlers per team. Needs 80-column printer, 32K Applesoft ROM.
Disk...$79.95
SUPER SOUND
Musical rhythms, gunshots, sirens, laser blasts, explosions...add these and many more exciting sounds to your Apple. Use them in your programs, or create your own SUPER SOUNDS. Needs 16K Applesoft.
Have a blast for only
$12.95...Tape
$16.95...Disk
ADD $2.00 U.S. $10.00 FOREIGN FOR SHIPPING
CALIFORNIA RESIDENTS ADD 6% SALES TAX
Open Tues. - Sun.
Don't see what you want here, then write or call today for your free catalog. We're saving one just for you.
RAINBOW COMPUTING INC.
GARDEN PLAZA SHOPPING CENTER
9719 RESEDA BOULEVARD DEPT. 1MI
NORTHRIDGE, CALIFORNIA 91324
PHONE (213) 349-0300
Visa/Mastercharge welcome.

MICRO

Dr. William R. Dial
438 Roslyn Avenue
Akron, Ohio 44320

6502 Bibliography: Part XXXI

890. The Cider Press (October, 1980)

Weiglin, Peter C., "Formatting," pg. 5-6.
Formatting and printing backwards on the Apple.

Bernheim, Phil, "Initialize New Files Automatically," pg. 6.
A handy tip for handling files kept on a periodic basis. For the Apple.

Bernheim, Phil, "Another Way to Find Hidden Characters," pg. 6.
A fix for hidden characters in Apple programs.

Bernheim, Phil, "Dates," pg. 8.
An easy way to encode and decode dates.

Hertzfeld, Andy, "Delete the Flashing Cursor," pg. 8.
A short monitor routine to silence the flashing cursor.

Bernheim, Phil, "Converting String Dates," pg. 11.
Routine to convert string dates to datum of the form MMDDYY.

Wilson, Gene, "Je M'Apple' PASCAL."
Several Pascal programs for the Apple.

891. The Apple Barrel 3, No. 7 (September/October, 1980)

Anon., "File Cabinet Partially Exposed," pg. 5-6.
A tutorial on the use of this important Apple utility.

Winter, Kevin, "Using the Backspace as a Delete Key," pg. 7.
A technique to add special functions to the Apple.

McGee, Pat, "Pascal Problems," pg. 9-13.
A number of problems with Apple Pascal and the fixes.

Barber, Bruce, "Screen Create," pg. 13-19.
A poor man's graphic tablet for the Apple.

Meador, Lee, "Disassembly of DOS 3.2," pg. 19-31.
The fifth installment of this excellent series, on the Apple DOS 3.2.

892. Stems from Apple 3, Issue 10 (October, 1980)

Sittel, Randy, "Are You Overtaxing Your Computer?", pg. 3.
A table of the current requirements of various Apple peripherals.

Ward, Dennis, "Display and Dec/Hex Converter," pg. 5.
Two short programs for the Apple.

Anon., "Kent's Puzzle Entries," pg. 8-11.
A series of 18 short programs for the Apple.

Todd, Allen W., "Blockedit," pg. 13-20.
An Apple Pascal disk utility, allowing examination and editing of any diskette block.

893. The Apple-Dillo (October, 1980)

Sethre, Tom, "Roots," pg. 3.
Another installment of this tutorial on Apple machine language, dealing with handling keyboard input.

Teas, George, "Pascal Primer," pg. 5.
A new column to help novice Pascal users with helpful programming hints.

Huffman, David, "Personalizing Your Catalog Heading," pg. 5.
An Apple DOS 3.2.1 tutorial.

Bartley, David, "Getting There Faster in Applesoft BASIC," pg. 6-7.
Speed up your Applesoft programs.

Sethre, Tom, "The Official Computer Game Player Cheater's Guide," pg. 7-8.

Sethre, Tom, "Memory Display," pg. 9.
A machine language routine for the Apple.

894. The G.R.A.P.E. Vine (September, 1980)

Sander-Cederlof, "One-Liner," pg. 1.
A one-liner for the Apple.

Anon., "Hi-Res Plotting of Characters from Character Table," pg. 2.
A Hi-Res writing program.

Anon., "Faster Shell," pg. 8.
This Faster Shell Sort is approximately 25% faster than the Hibbard Sort appearing in the last issue.

895. The Seed 2, No. 10 (October, 1980)

Anon., "Apple Pi Conventions," pg. 7.
A listing to standardize title pages on donated programs—for the Apple.

896. Applications 3, No. 2 (August, 1980)

Camiller, Derek, "GET vs. INPUT," pg. 12.
Simplify your input statements and responses.

Webster, Ian, "Telecom, Modems and Other Mysteries," pg. 14-17.
All about Modem use with the Apple in Australia.

Webster, Ian, "Relocation of Apple Machine Code," pg. 20-25.
This program will relocate machine code programs.

Haines, Ron, "Some Notes on the UCSD Assembler," pg. 26-27.
Additional comments to augment skimpy documentation of the Apple Pascal system.

Fields, Randy and Thompson, C., "Shorten Disk Commands," pg. 28.
Significantly shorten the Apple Disk Operating commands.

897. Radio Electronics 51, No. 10 (October, 1980)

Staff, "Buyer's Guide to Home Computers," pg. 45-84.
Among others, covers Apple, PET, OSI and Atari micros, including peripherals and accessories.

898. Fort Worth Apple User Group Newsletter (FWAUG) 2, No. 2 (October, 1980)

McVay, Ray, "Ampersoft," pg. 2-4.
Three routines for machine language string manipulation to forestall garbage collection and quicken sorting.

Meador, Lee, "DOS 3.2 Disassembly—9," pg. 6-21.
This installment of this important series discusses the Output State Machine and the data area and tables of commands and error messages.

Matzinger, Bob, "Make a Hello a Binary File," pg. 23.
A tricky way to put the boot program in Binary, for the Apple.

899. Personal Computing 4, No. 10 (October, 1980)

Fischer, Mike, "Word Search," pg. 34-39.
An Apple program to solve those jumbled letter matrices with hidden words.

Lubar, David, "Problem Solving and Computers," pg. 49-50.
Use of the Apple in problem solving.

Gaylord, Sam, "Cash In On the Power of Pascal," pg. 52-53.
Pascal program on the "melt value" of silver coins.

Jones, Jeremy C., "A Computer Spelling Exercise," pg. 54-55.
An educational program for OSI computers.

Welsh, Dave, "Rating BORIS 2.5," pg. 78-90.
All about the 6502-based computer programs called BORIS 2.5/Sargon 2.5.

900. MICRO No. 29 (October, 1980)

Berger, Thomas R., "An OSI Cheep Print," pg. 7-12.
Hardware and Software to interface OSI systems with a printer.

Morris, Gary A., "PRINT USING for Applesoft," pg. 14-17.
Program permitting user-defined formatting of the output of Applesoft.

Kolbe, Werner, "Define Your Own Function Key on PET," pg. 19-20.
Program routine to call a commonly used BASIC function from the keyboard.

DeJong, Marvin L., "An Improved Morse Code Receive Routine and Interface," pg. 23-26.
Hardware and Software for the AIM 65.

Borton, David, "Undedicating A Dedicated Microcomputer," pg. 27-28.
Developing a time-shared process controller for the KIM-1.

MacDonald, R., "A 'Stop-On-Address' Routine for KIM."
A routine to debug more efficiently through the use of a trap to stop on a specified address.

Schultz, Robert, "Tiny Pilot Complemental (Co-Pilot)."
Modification to Tiny Pilot improving the 'Match' statement—for the SYM.

Boynton, G.R., "For Multiple File Tape Backups," pg. 36-38.
This PET utility takes the drudgery out of making multiple backup copies of cassette files.

Wright, Loren, "PET Vet," pg. 39.
Comments on PET documentation and literature references.

Burcher, P.E., "Biorythm: An AIM BASIC Programming Exercise," pg. 51-55.
Programs providing an interesting demonstration for using the AIM and its printer.

Moyer, LeRoy, "Hexadecimal Printer," pg. 57-58.
A simple Apple II program permitting the user to specify the limits within which he wants the disassembler to operate.

Mulligan, John P., "Programming with Pascal," pg. 59-63.
Tutorial and Sorting routine for Apple Pascal.

Ellis, Dawn E., "Cassette Label Program," pg. 65-67.
Let the computer do the dirty work printing labels for those tape cassettes. For the Apple.

Rowe, Mike (Staff), "The MICRO Software Catalog: XXV," pg. 73-75.
Sixteen new items of software for 6502 systems are reviewed.

Dial, Dr. William R., "6502 Bibliography: Part XXV," pg. 76-77.
Some 65 new references to the extensive 6502 literature.

901. PEEK(65) Vol. 10 (October, 1980)

Williams, Jim, "Cold Start/System Crash Recovery," pg. 4-5.
Recovering from system crashes caused by errant POKE statements on OSI micros.

Lewton, Maurice P., "C1P or Superboard II GET Routine," pg. 7.
A listing for OSI micros providing a 'GET' routine.

Criscione, J.R., Jr., M.D., "More on OS-65U Password File Protection," pg. 11.
Comments and listing for program security.

902. Ham Radio 13, No. 10 (October, 1980)

Skeen, W.S., "A CW Keyboard Using the Apple II Computer," pg. 60-62.
Program Listing and simple interface hardware for using the Apple II with your Amateur station.

903. The G.R.A.P.E. Vine (October, 1980)

Anon., "Automatic Free Space with Catalog," pg. 3.
Short modified routine for the booting of your disks.

Anon., "DOS Changes on Disks," pg. 4.
How to personify your disks.

Anon., "What is Theomatics?", pg. 5-7.
Discussion and two listings related to biblical literature.

904. Creative Computing 6, No. 10 (October, 1980)

McClure, Jim, "The C2-4P Ohio Scientific Computer," pg. 17-23.
A review of a new micro.

Lubar, David, "Sound Advice," pg. 26-29.
DAC's and Music boards for 6502 machines.

Ahl, David H., "Easy Writer," pg. 34.
A review of a word processor for the Apple.

Platt, Charles, "Bargain-Basement Word Processing," pg. 54-55.
Review of the WP6502 word processor useful for OSI micros, Apple, PET, Atari, etc.

Carpenter, Chuck, "Apple-Cart," pg. 162-167.
How to avoid 'extra ignored' messages, saving strings on tape, listing for AC control routine, etc.

Yob, Gregory, "Personal Electronic Transactions," pg. 168-172.
How to get lower case letters on a PET, program providing automatic repeat of keys, PET machine language tutorial and examples, etc.

Blank, George, "Outpost: Atari," pg. 174-177.
String array demonstration, precautions for input/output, printer interface, programmer's aid, etc.

905. Apple/Sass (October, 1980)

Burson, Dick, "Name that Tone!", pg. 9.
Tone generating routines for the Apple.

Anon., "Math Drill II," pg. 14.
An educational Apple program.

Golding, Val J., "Hidden REM Formatter."
Two listings for the Apple.

906. Softalk 1 (October, 1980)

Wagner, Roger, "Assembly Lines," pg. 3-4.
Everyone's machine language guide.

907. Dr. Dobb's Journal 5, Iss. 9, No. 49 (October, 1980)

Gordon, H.T., "Opcode Design: 9 or 16 Bits?", pg. 22-23.
Discussion of 6502 code and codes of other microprocessors.

908. The Harvest 2, No. 2 (October, 1980)

Peterson, Todd C., "A Free Remote Control for Your Apple."
Use a receiver, a transmitter, and a patch cord for a remote control set-up.

Anon., "Parallel vs. Serial Printer Boards," pg. 7.
A discussion of the merits of these two options.

Anon., "Need to Get a Text File to a Printer in Pascal?", pg. 8.
A pascal starter routine for the Apple.

Holle, Dave, "Physical and Logical Sectors of DOS 3.3," pg. 9.
Some information on the new Apple DOS system.

Robbins, Mike, "16 Sector Patch for Dan's Disk Utility," pg. 9.
Just two lines added to fix this one up for 16 sectors.

Alexander, Leon, "Leon's Bell," pg. 10.
A short machine program that toggles the bell upon an INPUT or GET statement on the Apple.

909. The Apple Shoppe 1, No. 7 (July/August, 1980)

Staff, "Pascal PEEK and POKE," pg. 7-8.
A short tutorial for Apple Pascal.

Jeske, Mark, "General Purpose Plotter Program," pg. 11-13.
An educational plotting program for Apple.

Staff, "Programming the Graphics Tablet," pg. 14-18.
A tutorial and a Hi-Res Labeling program for the Apple fitted with a Graphics Tablet.

Crouch, Bill, "Down to Business," pg. 18-20.
A better formatting routine and hints on developing a business application for the Apple.

Staff, "Spinterm Hi-Res Dump in Assembly," pg. 21-27.
An improved Apple program.

Hugard, James, "Printing in Pascal," pg. 27-29.
Some useful information for Pascal users.

910. O.S.I. Users Independent Newsletter No. 6 (October, 1980)

Curley, Charles, "65D," pg. 2-4.
Here starts a series of articles on the innards of 65D.

911. KB Microcomputing No. 46 (October, 1980)

Baker, Robert W., "PET-Pourri," pg. 12-14.
Proganal is a useful PET utility. Several other items of PET software are reviewed.

Owens, James, "Computerized Estate Planning," pg. 31-35.
An OSI program to take the tedium out of this chore.

Perdue, William H., "PET Mini Monitor," pg. 88-91.
A short PET routine makes creating and saving machine language programs a snap.

Shawcross, William E., "OSI in the Sky," pg. 102-104.
A mail labeling program for the OSI family.

Carlson, Edward H., "Speed Up Your BASIC Programs," pg. 128-130.
Hints for better programs on the OSI C2-4P.

Phillips, Terry Edward, "Whoa, Apple," pg. 132-133.
Two machine language routines to tighten the reins on galloping video displays.

Greenberg, Gary, "Betting on Old POKEy," pg. 180.
A PET program using the POKE command to enter graphics.

Bauers, Barton M., Jr., "A Roundoff Function in Applesoft," pg. 210.
Keep your numbers at a manageable length with this Apple function.

912. Byte 5, No. 10 (October, 1980)

Zimmermann, Mark, "Floptran-IV: A Tiny Compiler," pg. 196-285.
A utility for the PET.

Harrod, Dennette A., "The 6502 Gets Microprogrammable Instructions," pg. 282-285.
A hardware approach to adding 64 user-defined instructions to the 6502.

913. SoftSide 3, No. 1 (October, 1980)

Laurence, Matthew, "One Liner," pg. 4.
Short graphics program for the Apple.

Clark, Terry, "IMHOTEP," pg. 18-21, 82-84.
A game for the Apple.

Pelczarski, Mark, "SoftSide Database-Part 2," pg. 22-23.
Program for Atari and Apple. Part 2 of a series.

Bohlke, David, "World Series," pg. 26-28.
A game for Atari.

Garon, James, "Octadraw," pg. 41, 68.
A graphics utility for the Atari.

Ward, Dennis, "Interna-Maze," pg. 43, 64-65.
A maze game for the Apple. Footprints, aerial view and all.

Sandy, Herb, "Moonlanding," pg. 44-45.
A graphics program for Apple.

Bohlke, David, "Double Cannon," pg. 50.
A game in Atari graphics.

Ohlund, Mark A., "Strings and Things," pg. 60-61.
An article dealing with Atari graphics and comparison with TRS-80 systems.

Bouchard, Rich and Garon, James, "Atari One-Liners," pg. 61.
Two shsort programs for the Atari.

Truckenbrod, Joan, "Computer Graphics," pg. 66-67, 80.
Two Apple graphics programs—easily translated to Atari.

Garon, James, "Hidden Colors in Graphics 8," pg. 69, 74.
A new Atari concept and sample programs.

914. Abacus II 2, Issue 10 (October, 1980)

Britto, Arthur, "No DOS INIT," pg. 2.
This program removes the DOS from a diskette to allow more room on the diskette. For Apple.

Anon, "IAC Application Note: Pascal Long Integer Fix," pg. 3-4.
This program is designed to repair the library module LONG-INTEGER in Apple Pascal.

Anon., "Pascal Hi-Res LOAD/SAVE to Disk," pg. 4-5.
This Apple Demo creates a Hi-Res picture in Pascal, then saves it to disk. Then it is reloaded and displayed.

Anon., "IAC Application Note: Pascal Utility Program," pg. 8-9.
Program in Apple Pascal which reads from REMIN and writes to disk.

Anon., "IAC Application Note: Linefeed-Pascal Utility."
Program to set or defeat Linefeeds, Apple Pascal.

Yee, David R., "Catalorganizer," pg. 12-13.
Organize the catalog on Apple diskettes in alphabetical order.

Anon., "IAC Application Note: COMCARD-Pascal Utility," pg. 13-14.
Program in Apple Pascal to set up COMCARD parameters.

Yee, David R., "Mass Lock and Unlock," pg. 14.
An Applesoft to lock or unlock all the files on an Apple diskette at once.

Anon., "IAC Application Note: Program Foreign—Pascal Utility," pg. 15-18.
A major utility in Apple Pascal.

915. 73 Magazine No. 242 (November, 1980)

Erdei, Steven C., "PL Tones from a KIM-1," pg. 112.
A KIM program that will generate a square wave tone anywhere in the range of 191 Hz to 66 Hz.

916. Personal Computing 4, No. 11 (November, 1980)

Gillie, Michael D., "Printing Pictures from Your PET," pg. 52-54.
A PET program to gather the characters off of the PET screen and print them on the Commodore 2022 printer.

Lubar, David, "Microbiocide," pg. 79-80.
A discussion and hints for debugging programs on the Apple.

917. Byte 5, No. 11 (November, 1980)

Cesa, Louis, "Kinetic String Art for the Apple," pg. 62-63.
High resolution program for the Apple.

Sokol, Dan, "Three-Dimensional Graphics for the Apple II," pg. 148-154.
A novel program for Hi-Res Apple graphics.

Ramsdell, Robert E., "The Power of VisiCalc," pg. 190-192.
All about this interesting piece of business related software for the Apple.

918. The Paper (Summer, 1980)

Haluza, Doug, "Cross Referenced Memory Map," pg. 4-6.
Memory map comparing locations on old and new PET ROMs.

Haluza, Doug, "Machine Language is Faster Than You Think," pg. 7.
Some examples for the PET.

Batcher, Bill, "The Evolution of a Puzzle," pg. 8-9.
A tutorial on handling strings on the PET.

Haluza, Doug, "Machine Language is Still Faster Than You Think," pg. 10-11.
A machine language sort with assembly language routine and a BASIC Demo for the PET.

Comito, JoAnn, "Stringing Your PET Along," pg. 12-15.
Two related discussions of Strings with examples, including an expression analyzing string routine for the PET.

Fowler, James, "Assembly Language Programming: Part 1, pg. 16-17.
Part One of a series for the PET programmer.

Eisner, Gerry, "POKE a Border," pg. 17-18.
A utility for the PET, with examples.

Bressler, Ralph, "Moving Around the Screen," pg. 19.
A utility for the PET.

Comito, JoAnn, "Not If's, And's or But's," pg. 20-22.
A tutorial on the PET's If-Then statement.

Bressler, Ralph, "BASIC Does It Better," pg. 23-25.
Discussion of a number of details in PET BASIC.

Haluza, Doug, "Trouble-Shooting Your PET," pg. 26-28.
Some hints for reviving the dead PET.

Bressler, Ralph, "Time Passes Quickly," pg. 28-29.
A tutorial on the use of the PET timing routine.

Comito, JoAnn, "Writing That Good Educational Program," pg. 30-33.
Two related articles on writing PET educational programs, with examples.

Bressler, Ralph, "PET Files," pg. 36-39.
A tutorial for PET Tape Files and a sample listing.

Comito, JoAnn and Bressler, Ralph, "Merging Programs," pg. 40-41.
All about the PET merge system.

Haluza, Doug, "An 80 by 50 Plotting Routine," pg. 42.
A PET listing for a plotting program.

Haluza, Doug, "PEEKing at BASIC," pg. 42-43.
A utility for PET.

Bressler, Ralph, "PET's Round Off Problems," pg. 43, 48.

919. The Harvest 2, No. 3 (November, 1980)

Lyle, Guy A., "Float, Float, Float Your Point," pg. 1-5.
A tutorial on floating point representation on the Apple.

920. Nibble No. 6 (November, 1980)

Connolly, Rick, "P.I.P. II: PIP Goes Disco," pg. 9-13.
A disk version of P.I.P. (Personal Inventory Program).

Litwin, Larry M., "A Simplified Way For a Tiger to Eat Apple Pie," pg. 21-22.
An Apple graphics Pascal program outputting to a Paper Tiger.

Mottola, R.M., "Amper-Interpreter," pg. 27-44.
Add "Print-Using" to your Apple's Instruction Set.

Harvey, Mike, "Blocking Very Large Files," pg. 45-47.
A technique for file management on the Apple.

Figueras, John, "Roundoff!", pg. 47.
Round off decimal values to a manageable format on the Apple.

Reynolds, William III, "Finding the Slot Number," pg. 48.
How to write programs addressing interface cards on the Apple.

Harrell, Keith, "Pascal Pointers and Principles," pg. 51-52.
A new column for Apple pascal programmers.

Rogan, J.A., "Four-In-A-Row," pg. 55.
A Lo-Res graphics game for the Apple.

Crossman, Craig, "An Assembly Language Tutorial," pg. 57.
This installment discusses the Screen Clear, RDKEY, and COUT in the Apple monitor.

Crossman, Craig, "Apple Tricks," pg. 59.
More Apple tricks including Fast DOS, Special Characters, and Unstable programs.

921. Iridis 2 (November, 1980)

Staff, "Fontedit," pg. 2-22.
A program to allow the Atari user to design character sets. Listing and complete information on the design and use of the program.

Staff, "Knotwork," pg. 22-36.
How to design a type of manuscript illumination in Atari programs. Listing and complete description.

Staff, "Hacker's Delight," pg. 36-46.
A compendium of useful memory locations for the Atari operating system.

Staff, "Loadfont," pg. 46-49.
A utility for use in writing programs on the Atari using private fonts.

922. KB Microcomputing No. 47 (November, 1980)

Smith, Wayne D., "A Mini Logic Monitor and Single-Cycler for Hardware Debugging," pg. 59-66.
Hardware for your KIM-1.

Bugg, Michael L., "Tinkering with Tiny BASIC," pg. 88-96.
Add four new and useful commands to Tiny BASIC as implemented on the KIM-1.

Brock, Thomas D., "Hard Copy for Apple Graphics," pg. 100-102.
Software for printing the high-resolution screen using a Diablo printer.

Bruey, Alfred J., "Microcomputer Hardware for the Handicapped," pg. 173-174.
Single-key data entry for the PET.

923. Recreational Computing 9, No. 3, Issue 48 (November/December, 1980)

Berggren, Stephen R., "DOZO in Pascal," pg. 32-36.
A Japanese DOZO game in Apple Pascal.

924. Creative Computing 6, No. 11 (November, 1980)

Guerard, Michael P., "Another Slice of Pi," pg. 8.
A way around a minor bug in arcsine and arccosine functions in Applesoft and TRS-80 Level II BASIC.

Piele, Donald T., "How To Solve It—With the Computer," pg. 66-71.
Part 3 of this series discusses Apple graphics and FOR—NEXT statements.

Kielian, Greg, "Bombproof Data Entry," pg. 102-104.
Unusual data entry methods for the Apple.

Lubar, David and Forsen, Richard, "Car Race: Anatomy of a Translation," pg. 118-119.
Translation for the Apple from a program written for a DEC PDP/11.

Yob, Gregory, "Personal Electric Transactions," pg. 164-169.
PET music programs, hardware for blinking lights with the PET, and a video car driver game.

Blank, George, "Outpost: Atari," pg. 170-171.
Atari Pascal is on the way, Atari Visicalc is available, also description of the PIA, ANTIC, CTIA, and POKEY chips for Atari.

Carpenter, Chuck, "Apple-Cart," pg. 172-180.
Absolute indexing, Indirect indexing, Interrupts, etc. on the Apple.

925. MICRO No. 30 (November, 1980)

Williams, Richard, "How to Use the Hooks," pg. 7-9.
Hooks allow the user to break the normal flow of control and redirect the Apple to his own routine.

Hart, John E., "An Ultra-Fast Tape Storage System," pg. 11-14.
A simple hardware modification to the OSI Superboard and a good home tape recorder yield data-transfer rates of up to 9600 baud.

Sebra, Randy, "SYM-Bell," pg. 17-24.
Use your SYM as a telephone memory dialer.

Morse, P. Kenneth, "Self-Modifying PET Programs," pg. 29-31.
A tutorial on writing a self-modifying program.

Needleman, Ted, "General Ledger for the Apple II," pg. 33-34.
The SBCS general ledger is a major business program for the Apple II. A review.

Froelich, Jerry W., M.D., "Microprocessors in Medicine: the 6502," pg. 36.
How the 6502 microprocessor is able to perform tasks in medical education nearly as well as large computer systems.

Morris, E.D., Jr. and Finkbeiner, Tim, "Ohio Scientific Users: Stop Those S⌐ Errors," pg. 37-39.
Correct the BASIC error message output, put out messages of your own, etc.

Allen, David P., "A Versatile Hi-Res Function Plotter for the Atari 400 and 800," pg. 47-50.
Discussion and listing of an Atari color graphics program.

Bridge, Theodore E., "John Conway's Game of Life Using Display Devices with Automatic Scrolling," pg. 53-58.
A KIM listing easily adapted to other 6502-based micros.

Peterson, Craig, "Step and Trace for the Apple II Plus," pg. 61-63.
Restore the Step and Trace functions of the original Apple II.

Flynn, Christopher J., "AIM 65 File Operations: Writing Text Files with BASIC," pg. 65-70.
The value of AIM BASIC is greatly enhanced with this technique of writing text files.

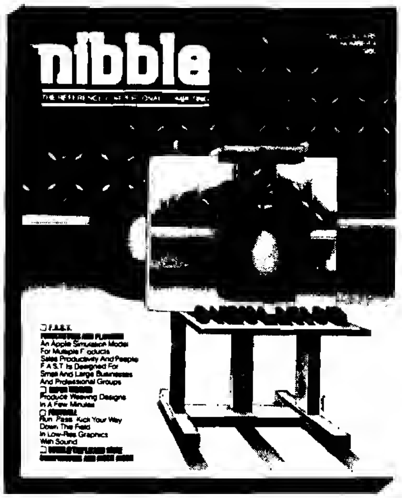
Rowe, Mike (Staff), "The MICRO Software Catalog: XXVI," pg. 72.
Review of five new 6502 software items.

Dial, William R., "6502 Bibliography: Part XXVI," pg. 76-77.
Some fifty-five new references to the 6502 literature.

ADVERTISERS' INDEX

April 1981

"NIBBLE® IS TERRIFIC" (For Your Apple)



NIBBLE IS: *The Reference for Apple computing!*

NIBBLE IS: One of the Fastest Growing new Magazines in the Personal Computing Field.

NIBBLE IS: Providing Comprehensive, Useful and Instructive Programs for the Home, Small Business, and Entertainment.

NIBBLE IS: A Reference to Graphics, Games, Systems Programming Tips, Product News and Reviews, Hardware Construction Projects, and a host of other features.

NIBBLE IS: A magazine suitable for both the Beginner and the Advanced Programmer.

Each issue of NIBBLE features significant new Programs of Commercial Quality. Here's what some of our Readers say:

- *"Certainly the best magazine on the Apple II"*
- *"Programs remarkably easy to enter"*
- *"Stimulating and Informative; So much so that this is the first computer magazine I've subscribed to!"*
- *"Impressed with the quality and content."*
- *"NIBBLE IS TERRIFIC!"*

In coming issues, look for:

☐ Numeric Keypad Construction Lab ☐ Assembly Language Programming Column
☐ Pascal Programming Column ☐ Data Base Programs for Home and Business
☐ Personal Investment Analysis ☐ Electronic Secretary for Time Management
☐ The GIZMO Business Simulation Game

And many many more!

NIBBLE is focused completely on the Apple Computer systems.

Buy NIBBLE through your local Apple Dealer or subscribe now with the coupon below.

Try a NIBBLE!

nibble No. 4

Box 325, Lincoln, MA. 01773 (617) 259-9710

I'll try nibble!
Enclosed is my $17.50 (for one year).
☐ **check** ☐ **money order**

(Please allow 4 to 6 weeks for delivery of 1st issue)
BACK ISSUES of NIBBLE are available for $2.00 + .50 postage and handling.

Name ____________________

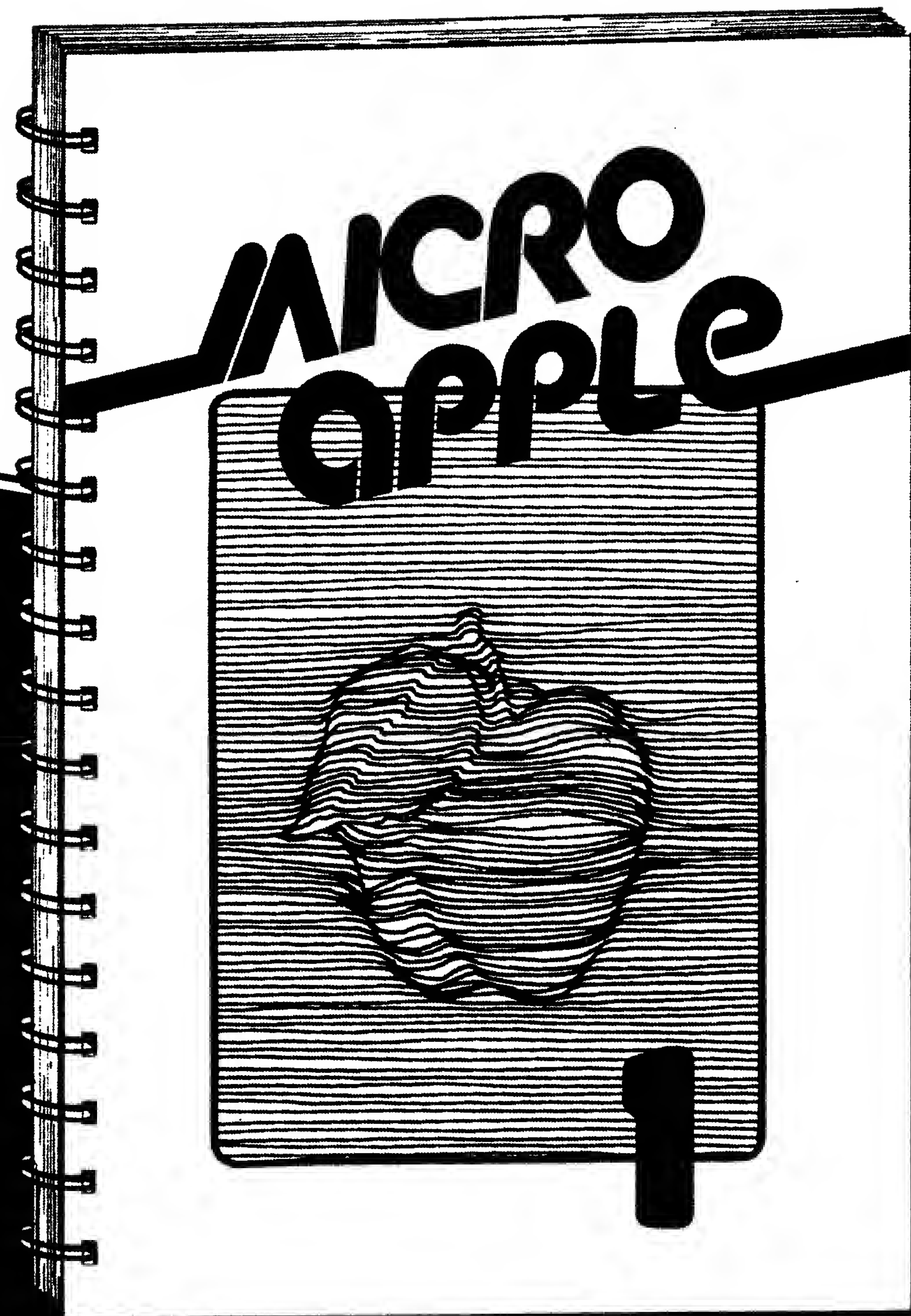
Address ____________________

City ____________________

State ____________ Zip ____________

NOTE:
First Class or Air Mail is required for all APO, FPO and all foreign addresses with the following additional amounts.
—USA, Canada, Mexico, APO, FPO $7.50
—Central and South America $9.00
—Europe $12.00
—Asia and elsewhere $15.00

$24.95 With diskette!



MICRO/Apple 1

This first volume in our new series contains 30 articles selected from MICRO, 1977-1980, updated by the authors or our staff. The MICRO staff has added introductory material and re-entered, listed, and tested the programs and put them on diskette.

Every user will want this highly practical work next to his Apple, with its chapters on BASIC Aids, Graphics, Education, Games, I/O Enhancements, Runtime Utilities, and References.

Get MICRO/Apple 1 at your local computer

224-page book and

More Than 30 Programs on Diskette — For Less Than $1.00 Apiece!

No Need to Type in Hundreds of Lines of Code!

MICRO™

P.O. Box

The home computer you thought was years away is here.

C8P DF

Ohio Scientific's top of the line personal computer, the C8P DF. This system incorporates the most advanced technology now available in standard configurations and add-on options. The C8P DF has full capabilities as a personal computer, a small business computer, a home monitoring security system and an advanced process controller.

Personal Computer Features
The C8P DF features ultra-fast program execution. The standard model is twice as fast as other personal computers such as the Apple II and PET. The computer system is available with a GT option which nearly doubles the speed again, making it comparable to high end mini-computer systems. High speed execution makes elaborate video animation possible as well as other I/O functions which until now, have not been possible. The C8P DF features Ohio Scientific's 32 x 64 character display with graphics and gaming elements for an effective resolution of 256 x 512 points and up to 16 colors. Other features for personal use include a programmable tone generator from 200 to 20KHz and an 8 bit companding digital to analog converter for music and voice output, 2-8 axis joystick interfaces, and 2-10 key pad interfaces. Hundreds of personal applications, games and educational software packages are currently available for use with the C8P DF.

Business Applications
The C8P DF utilizes full size 8″ floppy disks and is compatible with Ohio Scientific's advanced small business operating system, OS-65U and two types of information management systems, OS-MDMS and OS-DMS. The computer system comes standard with a high-speed printer interface and a modem interface. It features a full 53-key ASCII keyboard as well as 2048 character display with upper and lower case for business and word processing applications.

Home Control
The C8P DF has the most advanced home monitoring and control capabilities ever offered in a computer system. It incorporates a real time clock and a unique FOREGROUND/BACKGROUND operating system which allows the computer to function with normal BASIC programs at the same time it is monitoring external devices. The C8P DF comes standard with an AC remote control interface which allows it to control a wide range of AC appliances and lights remotely without wiring and an interface for home security systems which monitors fire, intrusion, car theft, water levels and freezer temperature, all without messy wiring. In addition, the C8P DF can accept Ohio Scientific's Votrax voice I/O board and/or Ohio Scientific's new universal telephone interface (UTI). The telephone interface connects the computer to any touch-tone or rotary dial telephone line. The computer system is able to answer calls, initiate calls and communicate via touch-tone signals, voice output or 300 baud modem signals. It can accept and decode touch-tone signals, 300 baud modem signals and record incoming voice messages. These features collectively give the C8P DF capabilities to monitor and control home functions with almost human-like capabilities.

Process Controller
The C8P DF incorporates a real time clock, FOREGROUND/BACKGROUND operation and 16 parallel I/O lines. Additionally a universal accessory BUS connector is accessible at the back of the computer to plug in additional 48 lines of parallel I/O and/or a complete analog signal I/O board with A/D and D/A and multiplexers.

Clearly, the C8P DF beats all existing small computers in conventional specifications plus it has capabilities far beyond any other computer system on the market today.

C8P DF is an 8-slot mainframe class computer with 32K static RAM, dual 8″ floppies, and several open slots for expansion.

Prices start at under $3,000.

Computers come with keyboards and floppies where specified. Other equipment shown is optional.

For literature and the name of your local dealer, CALL 1-800-321-6850 TOLL FREE.

OHIO SCIENTIFIC
a M/A-COM Company
1333 SOUTH CHILLICOTHE ROAD
AURORA, OH 44202 • (216) 831-5600